Device Storage Model White Paper

DSP0137 ***Status: Preliminary***

CIM Device Model White Paper on Storage

CIM Version 2.5

Abstract

This document describes the Logical Devices associated with data storage. A Logical Device provides an abstraction or emulation of a hardware entity for management purposes, has a distinct function, and participates in providing or implementing Services and Service Access Points. This paper discusses the storage concepts in CIM Schema versions 2.2 – 2.5.

Change History

Version 0.1		Initial release
Version 0.2		Updated to reflect feedback from the DMTF Sys/Dev working group.
Version 0.3	10/19/2000	Updated to reflect feedback from the SNIA DRM working group.

Table of Contents

1. Overview

Data storage Logical Devices are grouped into the following categories: Storage Device, Storage Medium, Storage Media Library, Fibre Channel, Defects, Data Organization, and Data Transport. Storage Media Library is covered in the Storage Media Library white paper. Fibre Channel is covered in the Fibre Channel white paper, and all remaining categories are covered in this white paper.

The Storage Device category represents the realization of the physical media players (i.e. Hard Disk Drive, CD-ROM player, etc). They are represented in the model as MediaAccessDevices. They describe the logical characteristics of the device.

The Storage Medium category represents the realization of the physical medium (i.e. disk, CD-ROM, tape, floppy, etc.). They are represented in the model as StorageExtents, which describe the characteristics of a storage space. They are also involved with the composition of several storage spaces into a new data storage space or the partitioning of a large medium into multiple data storage spaces.

The Defect category covers device errors as well as the mapping of "bad" blocks from the media.

The Data Organization category at this level does not involve file systems or files; it does involve the concepts of sparing, redundancy, and disk groups.

The Data Transport category represents the manageable aspects of the data transfer controllers and protocols.

2. Storage Device Objects and Associations

The Storage Device category represents the realization of the physical media players (i.e. Hard Disk Drive, CD-ROM player, etc). They are represented in the model as MediaAccessDevices. They describe the logical characteristics of the device. A MediaAccessDevice can have three distinct types of relationships. It can have a relationship to the media it relies on for data storage, to the data transport mechanism it utilizes, and to the physical objects that it is realized from.

- The **CIM_MediaPresent** association is used to represent the relationship to the media. MediaPresent is sub-classed from CIM_Dependency. This relationship shows which StorageExtents this MediaAccessDevice can access. If the device supports removable media and the media is not present, then this association would not be instantiated for that device. MediaPresent has a 'many to many' cardinality. A MediaAccessDevice can have access to multiple StorageExtents, and multiple MediaAccessDevices can access a StorageExtent.

- The **CIM_ControlledBy** association is used to represent the relationship to the data transport. This relationship is described in the Data Transport section.

- The **CIM_Realizes** association is used to represent the relationship between a LogicalDevice and a PhysicalElement. CIM_Realizes has a 'many to many' cardinality. A PhysicalElement can realize multiple LogicalDevices, and multiple PhysicalElements can realize a LogicalDevice.

  At a minimum a storage device "player" should be instantiated as a PhysicalPackage, and the media should be instantiated as PhysicalMemory. A realizes association would exist between the PhysicalPackage and the MediaAccessDevice to show the relationship between the physical package and the player. A realizes association would exist between the PhysicalMedia and the StorageExtent to show the relationship between the physical package and the "media". See the Storage Medium Associations sections for more detail on the latter relationship. For non-removable devices an optimization could be made, the same PhysicalPackage could realize both the "player" and "media" functionality. In addition the PhysicalPackage could model its connectors as PhyscialConnectors. The aggregate association ConnectorOnPackage is used to show the relationship between the connector and the package.

2.1. CIM_MediaAccessDevice

The MediaAccessDevice class tries to catch the characteristics and manageability of all types of media access devices. For example the property maximum data transfer rate applies to all types of storage devices and is defined in the MediaAccessDevice class. However, the property EOTWarningZoneSize only applies to a tape device and is defined in the TapeDevice sub-class.

The type of media access device is not modeled as an attribute. Rather, this "generic" class was defined along with several sub-classes. Although this can lead to a lot of duplicated data, it is flexible and extensible. Device "types" can change incrementally, which make standard type information very dynamic.

The following is the current list of MediaAccessDevice subclasses:

- **CIM_CDROMDrive** - This sub-class includes CDROM reader and writer drives.
- **CIM_DVDDrive** – This sub-class includes all of the types of DVD reader and writer drives.
- **CIM_DiskDrive** - This sub-class includes all hard disk drives, non-removable and removable (i.e. Jazz, Zip, etc.). There is a boolean property within MediaAccessDevice that indicates whether the device supports removable media. This sub-class only models the reader/writer properties of disk drives the media properties are modeled by StorageExtents.
- **CIM_DisketteDrive** – This sub-class includes all floppy drives.
- **CIM_MagnetoOpticalDrive** – This sub-class includes all MO drives.
- **CIM_TapeDrive** – This sub-class includes all Tape drives.
- **CIM_WORMDrive** – This sub-class includes all WORM drives.

3. Storage Medium Objects and Associations

The Storage Medium category represents the realization of the physical medium (i.e. disk, CD-ROM, tape, floppy, memory, etc.). They are represented in the model as StorageExtents, which describe the characteristics of a storage space. They are also involved with the composition of several storage spaces into a new data storage space or the partitioning of a large medium into multiple data storage spaces.

The first attempt at modeling RAID only considered the model defined by the DMTF Mass Storage Working Group's standard MIF group definitions which is based on the SCC model. However, it was realized that this model was not widely used for management. A need for a simpler model was identified. Therefore, the concept of a storage extent is very abstract. It allows for both a SCC and a simpler model to be represented very similarly. In addition, there are several aspects of data storage other than RAID that need to be modeled.

A StorageExtent can have several distinct relationships. It can have a relationship to the MediaAccessDevice, to its physical objects, to another StorageExtent, to the file system, to a defect, to a spare group, or to a redundancy group.

- The **CIM_MediaPresent** association is used to represent the relationship to the MediaAccessDevice. This relationship was described above.

- The **CIM_RealizesExtent** association is used to represent the relationship between the StorageExtent and its physical medium. RealizesExtent is an association between PhysicalComponent and StorageExtent. Normally, a StorageExtent would be realized by one of two different sub-classes of PhysicalComponent, PhysicalMedia or PhysicalMemory. RealizesExtent has '0 or 1 to many' cardinality. If realized, a StorageExtent can only exist on 1 PhysicalComponent. However, a PhysicalComponent, can realize several StorageExtents.

- The **CIM_BasedOn** association is used to represent the relationship of a StorageExtent to another Storage Extent. BasedOn is sub-classed from CIM_Dependency. This relationship shows the layering that exists between various StorageExtents. BasedOn has a 'many to many' cardinality. A StorageExtent can be formed from multiple StorageExtents, and multiple StorageExtents could use a StorageExtent.

  Several sub-classes of BasedOn were identified as a key relationship. These relationships are described below with their appropriate StorageExtent sub-class.

- The **CIM_ResidesOnExtent** association is used to represent the relationship of a FileSystem residing on a StorageExtent. ResidesOn is sub-classed from CIM_Dependency. This represents the relationship between the file system and the StorageExtents. ResidesOn has a 'many to many' cardinality. A file system can reside on multiple StorageExtents. Typically, a StorageExtent is used by a single file system. However, for flexibility and extensibility a StorageExtent can be used by multiple file systems. This relationship allows a file system to be based on any sub-class of StorageExtent, this definition allows for special cases and expansion. However, in a typical environment a file system ResidesOn a LogicalDisk.

- The **CIM_StorageDefect** association is used to represent the relationship of a StorageExtent to a defect. This relationship is described in the Data Organization section.

- The **CIM_ActsAsSpare** association is used to represent a sparing relationship between StorageExtents. This relationship is described in the Data Organization section.

- The **CIM_RedundancyComponent** association is used to represent the relationship of a StorageExtent to the redundancy group. This association is described in the Data Organization section.

3.1. CIM_StorageExtent

A reference in this document to StorageExtent applies to the super-class and any of the sub-classes. A reference to the super-class StorageExtent applies only to the super-class itself.

The various sub-classes of StorageExtent were created for specific purposes. However, their definitions were left somewhat vague to allow for special cases and expansion. The following specific purposes should be followed as often as possible in order to have a model that is consistently understandable and meaningful.

The layering of StorageExtents is described in the Examples section.

3.1.1. CIM_StorageExtent "super-class"

The StorageExtent class represents a logically contiguous region of storage medium. A StorageExtent could participate in a StorageRedundancyGroup and be used as the building blocks for representing RAID. This is not an abstract class and can be instantiated.

3.1.2. CIM_LogicalDisk

The LogicalDisk class represents a StorageExtent that is identifiable by a file system. For example the DeviceID field would contain a drive letter in a Windows environment, an access path in a Unix environment, and a volume name in a NetWare environment.

LogicalDisk has four specialized BasedOn associations: CIM_LogicalDiskBasedOnPartition, CIM_LogicalDiskBasedOnVolume, CIM_LogicalDiskBasedOnVolumeSet, and CIM_LogicalDiskBasedOnExtent. These specialized relationships were created to change the cardinality, extend the properties, and allow for specialized queries. A logical disk may be based on a disk partition. However, a logical disk might alternatively be based upon a StorageVolume, VolumeSet, or StorageExtent.

- The **CIM_LogicalDiskBasedOnExtent** association is used to represent the relationship between a LogicalDisk and its underlying StorageExtent. LogicalDiskBasedOnExtent has a 'many to 0 or 1' cardinality. A StorageExtent can participate in multiple LogicalDisks. A LogicalDisk can be based on 0 or 1 StorageExtent. Use the appropriate subclass of this association if the StorageExtent is a subclass.

- The **CIM_LogicalDiskBasedOnPartition** association is used to represent the relationship between a LogicalDisk and its underlying DiskPartition. LogicalDiskBasedOnPartition is sub-classed from LogicalDiskBasedOnExtent. If a LogicalDisk spans multiple DiskPartitions, then the LogicalDisk is actually based on a StorageVolume. LogicalDiskBasedOnPartition has a 'many to 0 or 1' cardinality. A DiskPartition can participate in multiple LogicalDisks. A LogicalDisk can be based on 0 or 1 Disk Partition.

- The **CIM_LogicalDiskBasedOnVolume** association is used to represent the relationship between a LogicalDisk and its underlying StorageVolume. LogicalDiskBasedOnVolume is sub-classed from LogicalDiskBasedOnExtent. LogicalDiskBasedOnVolume has a 'many to 0 or 1' cardinality. A StorageVolume can participate in multiple LogicalDisks. A LogicalDisk can be based on 0 or 1 StorageVolume.

- The **CIM_LogicalDiskBasedOnVolumeSet** association is used to represent the relationship between a LogicalDisk and its underlying VolumeSet. LogicalDiskBasedOnVolumeSet is sub-classed from LogicalDiskBasedOnExtent. LogicalDiskBasedOnVolumeSet has a 'many to 0 or 1' cardinality. A VolumeSet can participate in multiple LogicalDisks. A LogicalDisk can be based on 0 or 1 VolumeSet.

The model does not currently envision realizing a LogicalDisk directly from a physical class. Therefore, LogicalDisk has no specialized Realizes association.

3.1.3. CIM_DiskPartition

The DiskPartition class represents a StorageExtent that has been formatted for use by a specific type of file system. Typically, self-recognizable information has been placed on the storage medium. It has consumed part of the underlying storage space for this purpose. Usually, a disk partition should be based upon a StorageVolume. However, a DiskPartition may be realized directly from physical media.

DiskPartition has a specialized BasedOn association CIM_DiskPartitionBasedOnVolume and a specialized Realizes association CIM_RealizesDiskPartition.

- The **CIM_DiskPartitionBasedOnVolume** association is used to represent the relationship between a DiskPartition and its underlying StorageVolume. DiskPartitionBasedOnVolume has a 'many to 0 or 1' cardinality. If a DiskPartition is based on a StorageVolume, it can only be based on 1 StorageVolume. Multiple DiskPartitions can be based on a StorageVolume.

- The **CIM_RealizesDiskPartition** association is used to represent the relationship between a DiskPartition and its physical medium. RealizesDiskPartition is sub-classed from CIM_RealizesExtent. This association may only be used when there is no meaningful information about the StorageExtent to manage. RealizesDiskPartition has a '0 or 1 to many' cardinality. If a PhysicalComponent realizes a DiskPartition, a DiskPartition can only exist on 1 PhysicalComponent. However, a PhysicalComponent, can realize several DiskPartition.

3.1.4. CIM_TapePartition

The purpose for TapePartition is similar to DiskPartition but is for tape devices versus disk devices. TapePartition is used to logically represent the StorageExtent for an entire tape or for a single partition on that tape. Many of the new tape devices have the ability to make one tape look like many tapes and this concept is modeled through the use of TapePartition. A tape that is not partitioned is assumed to have one partition.

TapePartition has a specialized BasedOn association CIM_TapePartitionOnSurface.

- The **CIM_TapePartitionOnSurface** association is used to represent the relationship between a TapePartition and its underlying StorageExtent. TapePartitionBasedOnSurface has a 'many to 0 or 1' cardinality. If a TapePartition is based on a StorageExtent, it can only be based on 1 StorageExtent. Multiple TapePartitions can be based on a StorageExtent.

3.1.5. CIM_StorageVolume

The StorageVolume class represents the transition of a StorageExtent from one modeling domain to another. For example a RAID cabinet could model its redundancy and striping by building up StorageExtents. When the RAID cabinet is ready to present its storage to the operating environment, it

would represent this last StorageExtent as the StorageVolume. Another example a drive vendor is actively modeling the storage space of the drive. When the drive vendor is ready to present its storage to the operating environment for a logical disk, the drive vendor would present a Storage Volume. Effectively, a Storage Volume or its subclasses define a boundary between management domains for StorageExtents. A StorageVolume or its sub-class should not participate in a StorageRedundancyGroup.

3.1.5.1. CIM_VolumeSet

The VolumeSet class represents a StorageExtent that is sub-classed from StorageVolume. A VolumeSet is only used for representing RAID. A VolumeSet stripes its address space across the address space of one or more ProtectedSpaceExtents or AggregatePSExtents. ProtectedSpaceExtents and AggregatePSExtents are the resultant StorageExtents of a StorageRedundancyGroup comprised of other StorageExtents. Volume sets shall not overlap one another.

The VolumeSet specialization is derived from the VolumeSet Logical Unit defined in the document ANSI/NCITS 318-1998, titled: "SCSI Controller Commands - 2 (SCC-2)", developed by the National Committee for Information Technology Standards and published in 1998. It also covers the concepts of a volume set as defined by the DMTF Mass Storage Working Group's Standard MIF Groups Definition.

The mapping of property fields between CIM and SCC, and CIM and DMI are described in the Mapping section.

VolumeSets have a specialized BasedOn association, CIM_VolumeSetBasedOnPSExtent.

- The **CIM_VolumeSetBasedOnPSExtent** association is used to represent the relationship between a VolumeSet and its underlying ProtectedSpaceExtent. VolumeSetBasedOnPSExtent is VolumeSetBasedOnPSExtent has a '1 to many' cardinality. If instantiated, a VolumeSet must be based on at least one ProtectedSpaceExtent. Multiple VolumeSets can use a ProtectedSpaceExtent.

3.1.6. CIM_PhysicalExtent

The PhysicalExtent class represents a StorageExtent that defines how StorageRedundancyGroup metadata is distributed within it. The storage space presented by the PhysicalExtent includes that metadata. Contrary to the implication of its name, a PhysicalExtent is logical and may be based on any single StorageExtent.

PhysicalExtent is derived from the Redundancy Group P_Extent Descriptor defined in Table 77 of the document ANSI/NCITS 318-1998, titled: "SCSI Controller Commands - 2 (SCC-2)", developed by the National Committee for Information Technology Standards and published in 1998. (Note that the P_Extent defined by Table 17 of the above SCC-2 document is modeled in CIM as a CIM_BasedOn association together with the Antecedent StorageExtent.) PhysicalExtent also covers the concepts of a physical extent as defined by the DMTF Mass Storage Working Group's Standard MIF Groups Definition.

PhysicalExtent has a specialized Realizes association CIM_RealizesPExtent.

- The **CIM_RealizesPExtent** association is used to represent the relationship between a PhysicalExtent and its physical medium. RealizesPExtent is sub-classed from CIM_RealizesExtent. This association may only be used when there is no meaningful information about the StorageExtent to manage. RealizesPExtent has a '0 or 1 to many' cardinality. If a PhysicalComponent realizes a PhysicalExtent, a PhysicalExtent can only exist on 1 PhysicalComponent. However, a PhysicalComponent, can realize several PhysicalExtent.

3.1.7. CIM_AggregatePExtent

The AggregatePExtent call represents a StorageExtent that is logically constructed from one or more non-contiguous PhysicalExtents that are all based on the same underlying StorageExtent and which all belong to

the same StorageRedundancyGroup. An AggregatePExtent summarizes the properties of a PhysialExtent. For example, it contains how many blocks of check data exist, put doesn't provide information about which blocks contain the check data. Therefore, several instances of PhysicalExtent may be collapsed into a single instance of an AggregatePExtent. Generally, the PhysicalExtents will not be instantiated and a single BasedOn or RealizesAggregatePExtent association will be used to make the connection to the underlying storage.

The AggregatePExtent is derived from the concept of an aggregate PhysicalExtent as defined by the DMTF Mass Storage Working Group's Standard MIF Groups Definition.

AggregatePExtent has a specialized Realizes association CIM_RealizesAggregatePExtent.

- The **CIM_RealizesAggregatePExtent** association is used to represent the relationship between an AggregatePExtent and its physical medium. RealizesAggregatePExtent is sub-classed from CIM_RealizesExtent. This association may only be used when there is no meaningful information about the StorageExtent to manage. RealizesAggregatePExtent has a '0 or 1 to many' cardinality. If a PhysicalComponent realizes an AggregatePExtent, an AggregatePExtent can only exist on 1 PhysicalComponent. However, a PhysicalComponent, can realize several AggregatePExtent.

3.1.8. CIM_ProtectedSpaceExtent

The ProtectedSpaceExtent class represents a StorageExtent that resulted from creating a StorageRedundancyGroup from multiple underlying PhysicalExtents and includes only the user addressable space (excluding metadata). A single ProtectedSpaceExtent can represent multiple PhysicalExtents only when these PhysicalExtents are logically contiguous and on the same underlying physical media.

ProtectedSpaceExtent is derived from the Volume Set PS_Extent Descriptor defined in Table 119 of the document ANSI/NCITS 318-1998, titled: "SCSI Controller Commands - 2 (SCC-2)", developed by the National Committee for Information Technology Standards and published in 1998. This PS_Extent Descriptor is not directly modeled by CIM. The ProtectedSpaceExtent also covers the concept of a protected space extent as defined by the DMTF Mass Storage Working Group's standard MIF groups definition.

The mapping of property fields between CIM and SCC, and CIM and DMI are described in the Mapping section.

ProtectedSpaceExtent has a specialized BasedOn association, CIM_PSExtentBasedOnPExtent.

- The **CIM_PSExtentBasedOnPExtent** association is used to represent the relationship between the ProtectedSpaceExtent and its' underlying PhysicalExtent. PSExtentBasedOnPExtent has a 'many to many' cardinality.

3.1.9. CIM_AggregatePSExtent

The AggregatePSExtent class represents a StorageExtent that resulted from creating a StorageRedundancyGroup from multiple underlying StorageExtents and includes only the user addressable space (excluding metadata). A single AggregatePSExtent can represent multiple StorageExtents only when these are on the same underlying physical media. An AggregatePSExtent will likely span many non-contiguous regions of the underlying storage and allows modeling of a ProtectedSpaceExtents in a summary format when detailed information about the user data placement is not essential.

AggregatePSExtent is derived from the concept of an aggregate ProtectedSpaceExtent as defined by the DMTF Mass Storage Working Group's standard MIF groups definition.

AggregatePSExtents have two specialized BasedOn associations, CIM_AggregatePSExtentBasedOnPExtent and CIM_AggregatePSExtentBasedOnAggregatePExtent.

- The **CIM_AggregatePSExtentBasedOnPExtent** association is used to represent the relationship between the aggregate ProtectedSpaceExtent and its' underlying PhysicalExtent. AggregatePSExtentBasedOnPExtent has a 'many to many' cardinality.

- The **CIM_AggregatePSExtentBasedOnAggregatePExtent** association is used to represent the relationship between the aggregate ProtectedSpaceExtent and its' underlying aggregate PhysicalExtent. AggregatePSExtentBasedOnAggregatePExtent has a 'many to many' cardinality.

3.1.10. CIM_Memory

The Memory class represents a StorageExtent that is realized by CIM_PhysicalMemory. This could be anything from a cache to a RAM Disk.

- The **CIM_RealizesExtent** association is used to represent the relationship between the Memory and its PhysicalMemory. RealizesExtent is an association between PhysicalComponent and logical memory. RealizesExtent has '0 or 1 to many' cardinality. If realized, Memory can only exist on 1 PhysicalMemory. However, PhysicalMemory, can realize several logical Memories.

There are a number of related memory associations.

- The **CIM_AssociatedMemory** association is used to represent the relationship between the Memory and the LogicalDevice where it is installed. AssociatedMemory has a 'many to many' cardinality.

- The **CIM_ComputerSystemMemory** aggregate association is used to collect Memory that is participating in the UnitaryComputerSystem. A UnitaryComputerSystem requires at least one unit of Memory.

- The **CIM_AssociatedProcessorMemory** association is used to represent the relationship between Memory and a Processor.

The following are sub-classes of Memory:

3.1.10.1. CIM_NonVolatileStorage

The NonVolatileStorage class represents Memory that is persistent (i.e. flash and EEPROM). This does not include battery backed up memory.

NonVolatileStorage has one specialized association.

- The **CIM_BIOSLoadedInNV** association is used to represent the relationship between a BiosElement and the NonVolatileStorage where it is contained.

3.1.10.2. CIM_VolatileStorage

The VolatileStorage class represents Memory that is not persistent. This storage can be battery backed up by having a battery associated with it via the CIM_AssociatedBattery association.

3.1.10.3. CIM_CacheMemory

The CacheMemory class represents Memory that is used as fast upper-level temporary storage in a processor hierarchy. This class does not represent a generic cache buffer.

3.1.11. CIM_Snapshot

The Snapshot class is an optional construct in the Storage Model. It represents storage that contains a full copy of another StorageExtent, or the changes to an Extent when performing a delta-based (delta before or delta after) copy. A Snapshot is sub-classed from StorageExtent and adds several properties. These

properties are related to whether the storage is synchronized with the original Extent, and the synchronization time.

Snapshot's definition in the CIM usage is not equivalent to the act of creating a volume or file-based snapshot or point in time copy. It is at a much lower level and only represents the use of storage to hold a copied image of a StorageExtent, or the changes to an Extent. Use of the Snapshot object when making a full copy is only recommended if the Extent's purpose is to describe the existence of a copy. (This use will be very infrequent.) The problem with describing an Extent solely as a "Snapshot" is that when the snapshot/copy is broken, the object must be destroyed. Typically, this object still holds valid data that can be manipulated as a StorageVolume or more generic Extent. Therefore, the object would have to be re-instantiated.

If the "Full Copy" object is to be treated as a StorageVolume or more general Extent, then it should be modeled as such (i.e., not as an instance of Snapshot but as an instance of a generic StorageExtent or StorageVolume). A new association in CIM V2.3 (CIM_Synchronization) is used to describe the fact that one object is synchronized with another. This association defines a generic model describing that one object (an Extent, a FileSystem, a LogicalFile, etc.) is synchronized with another. It does not address the underlying mechanisms to accomplish this synchronization. This approach solves the problem of instance deletion/recreation when a snapshot/copy is broken. Only the instance of CIM_Synchronization has to be deleted – not the referenced objects.

When describing a delta-based snapshot/point in time copy, the Snapshot object represents the intermediate buffer space (or cache) holding the before/after image changes to the original Extent. For example, when doing a "delta before" snapshot, the resultant target would be modeled as a StorageExtent that is BasedOn the original Extent and its changes (the CIM_Snapshot that holds these changes). This is described more fully in Example xxx later in this paper.

Snapshot has a specialized Dependency association.

- The **CIM_SnapshotOfExtent** association is used to represent the relationship between a delta data or copy and its source StorageExtent. It has one additional property, SnapshotType. SnapshotType indicates the type of snapshot, copy, delta before, delta after, and "other". SnapshotBasedOnExtent has a '0 or1 to many' cardinality. If a Snapshot exists, it can only be associated with one source Extent. A StorageExtent could have several copies or deltas defined (CIM_Snapshots).

4. Defect Objects and Associations

The model captures two types of error situations. The first type, CIM_DeviceErrorCounts, tracks error statistics for a LogicalDevice. The second type, CIM_StorageError, records the mapping of "bad" blocks from the media.

4.1.1. CIM_DeviceErrorCounts

DeviceErrorCounts is a collection of error counters for a device. DeviceErrorCounts is sub-classed from CIM_StatisticalInformation.

- The **CIM_ErrorCountersForDevices** association is used to represent the weak relationship between the DeviceErrorCounts and the LogicalDevice. ErrorCountersForDevices is sub-classed from CIM_Statistics. ErrorCountersForDevices has a '1 to many' cardinality. If the DeviceErrorCounts object is instantiated, then its' related LogicalDevice must be instantiated. An instance of DeviceErrorCounts can only be related to a single LogicalDevice. However, a LogicalDevice can have relationships with multiple instances of DeviceErrorCounts.

4.1.2. CIM_StorageError

The purpose of StorageError is to represent the blocks within a StorageExtent that have been marked bad.

- The **CIM_StorageDefect** association is used to represent a collection of weak relationships between the StorageError and the StorageExtent. StorageDefect has a '1 to many' cardinality. If the StorageError object is instantiated, then its' related StorageExtent must be instantiated. An instance of a StorageError can only be related to a single StorageExtent. However, a StorageExtent can have relationships with multiple instances of StorageErrors.

5. Data Organization Objects and Associations

Data organization explores the concepts of redundancy, grouping, & sparing.

5.1. CIM_RedundancyGroup

A RedundancyGroup collects a set of ManagedSystemElements together to create a new set of protected or striped ManagedSystemElements. This collection indicates that the aggregated components together provide redundancy. All elements in a RedundancyGroup should be of the same object class. Three specialized sub-classes of RedundancyGroup are described below. Use of the appropriate sub-classes is desired.

A RedundancyGroup has a distinct relationship, CIM_RedundancyComponent.

- The **CIM_RedundancyComponent** aggregate association is used to collect objects participating in the RedundancyGroup. RedundancyComponent is sub-classed from CIM_Component. This aggregate association establishes which objects is part of the collection. RedundancyComponent has a 'many to many' cardinality. A RedundancyGroup is a multitude of ManagedSystemElements. A ManagedSystemElement can be part of multiple RedundancyGroups.

The following are the specialized sub-classes of RedundancyGroup:

5.1.1. CIM_StorageRedundancyGroup

The StorageRedundancyGroup class is used to represent a RedundancyGroup that aggregates StorageExtents. The resultant StorageExtent provides some level of protection or striping.

The StorageRedundancyGroup is derived from the Redundancy Group Logical Unit defined in the document ANSI/NCITS 318-1998, titled: "SCSI Controller Commands - 2 (SCC-2)", developed by the National Committee for Information Technology Standards and published in 1998. It also covers the concepts of a redundancy group as defined by the DMTF Mass Storage Working Group's Standard MIF Groups Definition.

The StorageRedundancyGroup allows the level of redundancy to be "none". In this mode, it can be used to construct stripes (RAID 0) or concatenates the underlying StorageExtents. Use this method when concatenating or striping StorageExtents that are not ProtectedSpaceExtents. When striping or concatenating ProtectedSpaceExtents, a VolumeSet should be used, not a StorageRedundancyGroup.

StorageRedundancyGroup has two distinct relationships. However, for any one StorageRedundancyGroup only one of the two relationships should be instantiated. CIM_ExtentRedundancyComponent and CIM_PExtentRedundancyComponent aggregate associations are used as the collection mechanism for StorageRedundancyGroup.

- The **CIM_ExtentRedundancyComponent** aggregate association is used to collect StorageExtents participating in the StorageRedundancyGroup. ExtentRedundancyComponent is sub-classed from CIM_RedundancyComponent. A StorageRedundancyGroup is a multitude of StorageExtents. Typically, a StorageExtent would only participate in a single StorageRedundancyGroup. StorageRedundancyGroup could be used for modeling striping. It is conceivable that a StorageExtent could be multi-striped. In this case it would participate in multiple StorageRedundancyGroups. If a StorageExtent participates in multiple StorageRedundancyGroups it can not contain any metadata. ExtentRedundancyComponent has a 'many to many' cardinality.

- The **CIM_PExtentRedundancyComponent** aggregate association is used to collect PhysicalExtents participating in the StorageRedundancyGroup. PExtentRedundancyComponent is sub-classed from CIM_ExtentRedundancyComponent. A StorageRedundancyGroup is a multitude of PhysicalExtents. A PhysicalExtent can be part of multiple StorageRedundancyGroups. However, if a PhysicalExtent participates in multiple StorageRedundancyGroups it can not contain any metadata. PExtentRedundancyComponent has a 'many to many' cardinality.

- The **CIM_AggregateRedundancyComponent** aggregate association is used to collect AggregatePExtents participating in the StorageRedundancyGroup. AggregateRedundancyComponent is sub-classed from CIM_ExtentRedundancyComponent. A StorageRedundancyGroup is a multitude of AggregatePExtents. An AggregatePExtent can be part of multiple StorageRedundancyGroups. However, if a PhysicalExtent participates in multiple StorageRedundancyGroups it can not contain any metadata. AggregateRedundancyComponent has a 'many to many' cardinality.

5.1.2. CIM_ExtraCapacityGroup

The ExtraCapacityGroup is used to represent a RedundancyGroup that provides protection by having more than is needed of a type of ManagedSystemElement. These ManagedSystemElements are all active. One use would be to represent redundant controllers that are load balancing. Another use would be to represent a group of fans. The CIM_RedundancyComponent aggregate association is used to collect objects participating in the ExtraCapacityGroup. An ExtraCapacityGroup should consist of ManagedSystemElements of the same type.

5.1.3. CIM_SpareGroup

The SpareGroup is used to represent a RedundancyGroup that provides protection by having extra of a type of ManagedSystemElements. These ManagedSystemElements are usually not actively in use, but may be powered. A SpareGroup might represent a collection of drives and spare drives. The CIM_RedundancyComponent aggregate association is used to collect the type of ManagedSystemElement in use that is participating in the SpareGroup. A SpareGroup should consist of ManagedSystemElements of the same type.

SpareGroup has one specialized association:

- The **CIM_ActsAsSpare** association is used to represent the relationship of the spare to the SpareGroup, the collection of ManagedSystemElements it can spare. ActsAsSpare has a 'many to many' cardinality. A spare can be a spare from multiple SpareGroups. A SpareGroup can have multiple spares.

5.2. CIM_DiskGroup

A DiskGroup is a named collection of ManagedSystemElements that are intended to be DiskDrives and/or the StorageExtents accessed via those DiskDrives. DiskGroup is sub-classed from CIM_CollectionOfMSEs. The collection is used to limit the BasedOn relationships of the StorageExtents within the DiskGroup. This collection does not represent any redundancy. This is used for purposes of allocation. For example, a StorageVolume created "within the DiskGroup" is restricted to be completely made up of StorageExtents with underlying storage completely contained or realized from the DiskDrives within the DiskGroup. A DiskGroup can consist of DiskDrives, StorageExtents, and DiskGroups.

DiskGroup has three aggregate associations CIM_DriveInDiskGroup, ExtentInDiskGroup, and GroupInDiskGroup as the collection mechanism.

- The **CIM_DriveInDiskGroup** aggregate association is used to collect DiskDrives participating in the DiskGroup. DriveInDiskGroup is sub-classed from CIM_CollectedMSEs. DriveInDiskGroup has a '0 or 1 to many' cardinality. A DiskDrive can be part of zero or a maximum of one DiskGroup. A DiskGroup can consist of multiple DiskDrives.

- The **CIM_ExtentInDiskGroup** aggregate association is used to collect StorageExtents participating in the DiskGroup. ExtentInDiskGroup is sub-classed from CIM_CollectedMSEs. ExtentInDiskGroup has a '0 or 1 to many' cardinality. A StorageExtent can be part of zero or a maximum of one DiskGroup. A DiskGroup can consist of multiple StorageExtents.

- The **CIM_GroupInDiskGroup** aggregate association is used to collect DiskGroups participating in the DiskGroup. GroupInDiskGroup is sub-classed from CIM_CollectedMSEs. A DiskGroup may consist of smaller named collections, DiskGroups. GroupInDiskGroup has a '0 or 1 to many' cardinality. The smaller DiskGroup can be part of zero or a maximum of one larger DiskGroup. A larger DiskGroup can consist of multiple smaller DiskGroups.

6. Data Transport

Data Transport represents the manageable aspects of the data path. Typically, this information is only available from the hardware controller provider. Data transfer controllers are represented by CIM_Controllers. Data transfer protocols are represented by the CIM_ControlledBy relationship. The concept of connectivity is discussed in CIM_ProtocolEndpoints.

6.1. CIM_Controller

A Controller represents a class of hardware having an explicit protocol stack and existing primarily for communication with devices. The Controller object is defined as abstract, but subclasses (such as Serial, Parallel, USB, and SCSI) specify concrete and more specific constructs. The Controller class tries to catch the characteristics and manageability of all types of controllers.

Devices are related to Controllers using the ControlledBy association or one of its subclasses. Communication between these entities is for data access resets, and configuration requests. Protocol details are typically defined in the subclass for Controller itself or in subclasses of the ControlledBy association.

Controller or one of its' sub-classes is distinguished from NetworkAdapters or one of its' sub-classes by the fact the Controllers do not model their protocol stack. If there is a need to model the protocol stack of a Controller's protocol, then a DeviceIdentity relationship is used between the appropriate sub-class of Controller and the sub-class of NetworkAdapter.

The type of controller is not modeled as an attribute. Rather, this "abstract" class was defined along with several sub-classes. Although this can lead to a lot of duplicated data, it is flexible and extensible. Controller "types" can change incrementally, which make standard type information very dynamic.

- **DeviceIdentity** indicates that two LogicalDevices represent different aspects of the same underlying entity. This association refines the CIM_LogicalIdentity superclass by restricting it to the Device level and defining its use in well-understood scenarios. One of these scenarios is to represent that a Device is both a 'bus' entity and a "functional" entity. For example, a Device could be both a PCI Device (or a USB Device), as well as a CIM_Keyboard. The other scenario is where a Device plays multiple functional roles that can not be distinguished by their hardware realization alone. For example, a Fibre Channel adapter might have aspects of both a NetworkAdapter and a SCSIController.

A MediaAccessDevice has a ControlledBy relationship to a Controller.

- The **CIM_ControlledBy** association is used to represent the relationship between a Controller and a MediaAccessDevice. This association adds properties to, and is sub-classed from CIM_DeviceConnection. ControlledBy has a 'many to many' cardinality. For storage a Controller can access multiple MediaAccessDevices. A MediaAccessDevice can be accessed via multiple Controllers.

The following is a list of Controller subclasses that pertain to storage (i.e. CIM_VideoController is not listed). For a complete list of controllers see the Device MOF.

- **CIM_IDEController –** This sub-class is used to represent IDE controllers. This includes ATA and ATAPI controllers.

- **CIM_ManagementController** – This sub-class is used to represent controllers that are dedicated to perform a specific management function (i.e. SES, SAF-TE, and statistical gathering and analysis). This sub-class also includes controllers that are minimal capacity Asics (i.e. I2C).

- **CIM_PCIController –** This sub-class is used to represent PCI controllers.

- **CIM_PCMCIAController** - This sub-class is used to represent PCMCIA controllers.

- **CIM_ParallelController** – This sub-class is used to represent parallel controllers.

- **CIM_SCSIController** - This sub-class is used to represent a controller implementing SCSI commands. For example, a FibreChannelAdapter implementing SCSI commands could use the DeviceIdentity association between FibreChannelAdapter and SCSIController. SCSIController has a specialized relationship, CIM_SCSIInterface.
  - **CIM_SCSIInterface** is a specialized sub-class of CIM_ControlledBy. There is data unique to the relationship between the SCSI controller and SCSI device (i.e. Negotiated speeds). SCSIInterface has a 'many to many' cardinality. A SCSIController can access multiple MediaAccessDevices. A MediaAccessDevice can be accessed via multiple SCSIControllers.

- **CIM_SerialController** – This sub-class is used to represent serial controllers. SerialController has a specialized relationship, CIM_SerialInterface.
  - **CIM_SerialInterface** is a specialized sub-class of CIM_ControlledBy. There is data unique to the relationship between the controller and device. SerialInterface has a 'many to many' cardinality. A SerialController can access multiple MediaAccessDevices. A MediaAccessDevice can be accessed via multiple SerialControllers.

- **CIM_USBController –** This sub-class is used to represent USB controllers. The concept of USBController is described in the USB white paper.

6.2. CIM_ProtocolEndpoints

A ProtocolEndpoint is a communication point from which data may be sent or received and provides the ability to utilize or invoke a Service. If there is a need to model the protocol stack of a Controller's protocol, then a DeviceIdentity relationship is used between the appropriate sub-class of Controller and the sub-class of NetworkAdapter. NetworkAdapter have specialized ProtocolEndpoints associated with it. The concept of ProtocolEndpoints is described in the Network white paper.

7. Examples

The following examples depict typical simplistic instantiations of storage environments. They are presented here to help clarify the objects and relationship within the storage model.

7.1. SCSI Hard Drive

This is depicting an example of a SCSI controller on a PC motherboard that is controlling a single SCSI hard drive. A single LogicalDisk represents the device. The objects between the two magenta dashed lines are from the physical model. The objects above the first magenta line are logical objects. All objects, expect FileSystem, are from the device model. The FileSystem object is from the system model and is not described. The associations are described with the objects they pertain to.

7.1.1. Discussion of Figure 7-1: SCSI Hard Drive

- **Chip (Physical)–** The Chip represents the physical integrated circuit (IC). The IC may provide both PCI and SCSI functionality. In this example only the SCSI functionality is represented.

- The chip is physically on the card. This relationship is identified via the **ChipOnCard** association.
- The chip realizes the SCSI initiator functionality. This relationship is identified via the **Realizes** association.

- **Card (Physical) –** The Card represents the motherboard that the Controller IC is on. Typically, a motherboard would contain multiple types of chips. However, for this example we only care about the storage controller.
- The relationship between the storage controller chip and the motherboard is identified via the **ChipOnCard** association.

- **PhysicalPackage (Physical) –** The PhysicalPackage represents the SCSI hard drive.
- The PhysicalPackage realizes the "player" functionality of the drive via the **Realizes** association.
- The relationship between the drive and its' PhysicalMedia is represented via the **PackagedComponent** association. The optimization of having the PhysicalPackage realizes the StorageVolume via the RealizesExtent association is permitted. If this optimization were used, then the PhysicalMedia would not be instantiated.

- **PhysicalMedia (Physical) –** The PhysicalMedia represents the media (platters) of the SCSI hard drive.
- The PhysicalMedia realizes the storage functionality of the drive via the **RealizesExtent** association. The optimization of having the PhysicalMedia realizes the storage functionality of a DiskPartition versus a StorageVolume is permitted. Typically, this optimization would occur if there were no hardware providers.
- The relationship between the drive and its' PhysicalMedia is represented via the **PackagedComponent** association.

- **SCSIController (Logical)–** The SCSIController represents the SCSI functionality that the IC provides.
- The relationship between the functionality and the physical IC is identified via the **Realizes** association.
- The SCSIController accesses the SCSI hard drive. This relationship is represented via the **SCSIInterface** association between the SCSIController and DiskDrive.
- The SCSIController provides an interface for the Logical Unit to the operating environment, host. This relationship is represented via the **SCSIInterface** association between the SCSIController and StorageVolume.

- **DiskDrive (Logical) –** The DiskDrive represents the "player" functionality of the SCSI target.
- The PhysicalPackage realizes the "player" functionality of the drive via the **Realizes** association.
- The SCSI initiator/target relationship between the DiskDrive and the SCSIController is represented via the **SCSIInterface** association.
- The relationship between the "player" and the media is represented via the **MediaPresent** association.

- **StorageVolume (Logical) –** The StorageVolume represents the data store functionality of a SCSI logical unit.
- The PhysicalMedia realizes the StorageVolume functionality of the drive via the **RealizesExtent** association.
- The relationship between the "player", DiskDrive, and the data store, StorageVolume, is represented via the **MediaPresent** association.
- One or more DiskPartitions could be placed on the data store. The relationship between the StorageVolume and DiskPartition is represented via the **DiskPartitionBasedOnVolume** association. In this example, only one DiskPartition is placed on the data store. In a mainframe environment there may not be a DiskPartition. In this case the LogicalDisk would be BasedOn the StorageVolume via the LogicalDiskBasedOnStorageVolume association.
- The access to the StorageVolume, SCSI logical unit, via the operating environment, host, is represented by the **SCSIInterface** association between the SCSIController and StorageVolume.

- **DiskPartition (Logical) –** One or more DiskPartitions may be placed on the StorageVolume via a FDISK type operation. In this example only one DiskPartition is placed on the same DiskDrive. If multiple DiskPartitions were placed on the DiskDrive, then multiple instances of DiskPartition and its' associations would be instantiated. In a mainframe environment there may not be a DiskPartition. In that case DiskPartition would not be instantiated.

- The relationship between the StorageVolume and DiskPartition is represented via the **DiskPartitionBasedOnStorageVolume** association.
- A LogicalDisk may be placed on the DiskPartition. The relationship between the DiskPartition and the LogicalDisk is represented via the **LogicalDiskBasedOnPartition** association.

- **LogicalDisk (Logical) –** A file system gives a LogicalDisk an identifiable name. For example in a Windows environment, the name is the drive letter, in a Unix environment, the name is the access path, and in a NetWare environment the name is the volume name. A LogicalDisk can only be based on one DiskPartition.
- The relationship between the DiskPartition and the LogicalDisk is represented via the **LogicalDiskBasedOnPartition** association. In a mainframe environment there may not be a DiskPartition. In a mainframe environment there may not be a DiskPartition. In this case the LogicalDisk would be BasedOn the StorageVolume via the LogicalDiskBasedOnStorageVolume association.
- The relationship between the LogicalDisk and FileSystem is represented via the **ResidesOnExtent** association. A FileSystem may span multiple LogicalDisks.

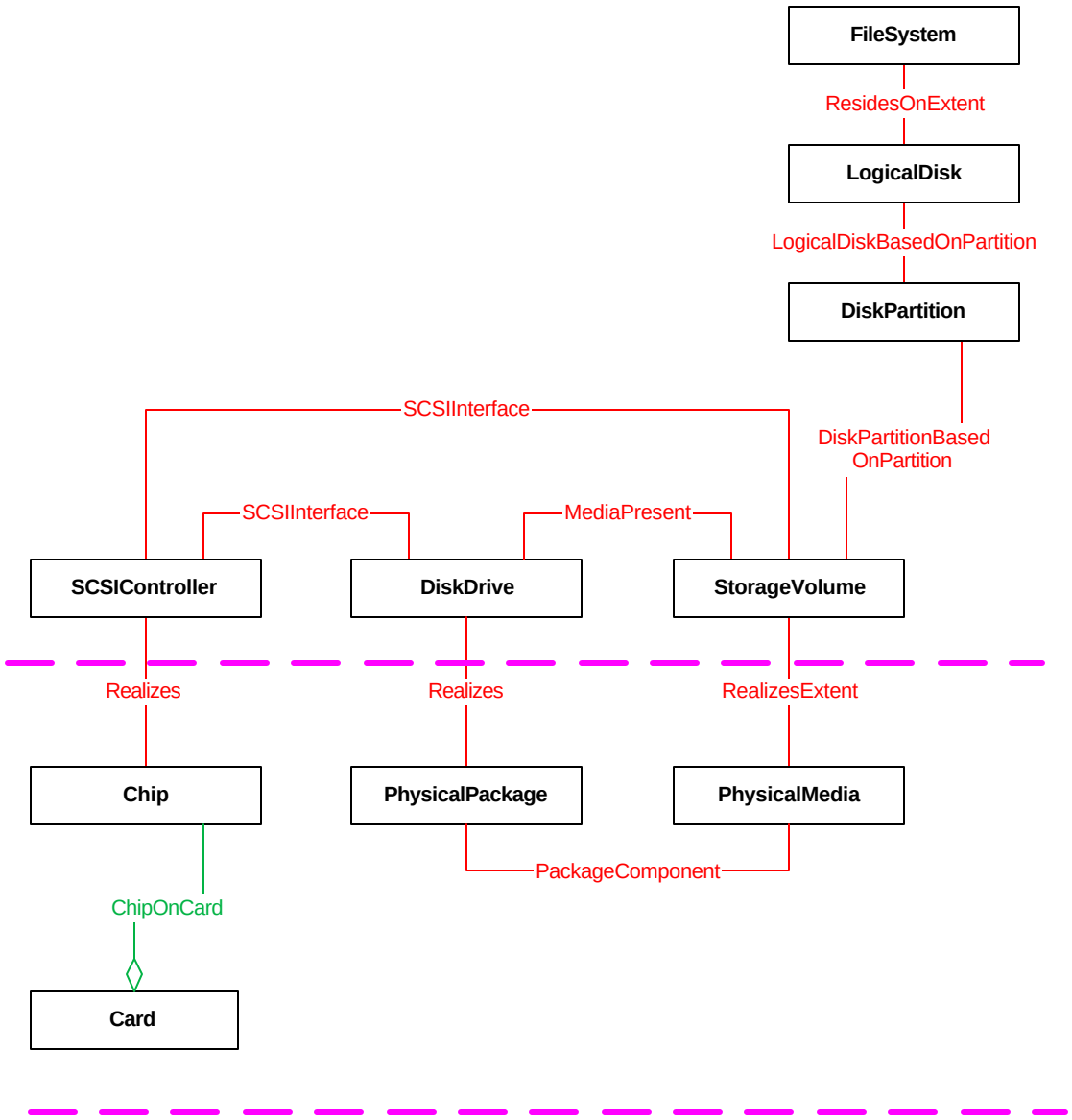

Figure 7-1: SCSI Hard Drive

7.2. External SCSI RAID-1 + 0

This example depicts a SCSI RAID cabinet connected to PC environment. The SCSI RAID cabinet consists of four SCSI hard drives in a RAID-1 + 0 configuration. A single LogicalDisk represents the device. The objects between the bottom two magenta dashed lines are the physical objects related to the RAID cabinet. The objects between the top blue line and the middle magenta line are the logical objects related to the RAID cabinet. The objects above the top blue line are the logical objects related to the PC environment or host. The objects in the two lower magenta boxes (dash dot) represent the mirroring of data (RAID 1). The objects in the upper magenta box (dash dash dot) represent the striping of data (RAID 0). All physical objects are from the physical model. All logical objects, expect FileSystem, are from the device model. The FileSystem object is from the system model and is not described. The associations are described with the objects they pertain to.

7.2.1. Discussion of Figure 9-2 SCSI RAID-1 + 0 Cabinet

- **Chip (Physical)–** The Chip represents the physical integrated circuit (IC). In this example the IC is capable of supporting two functionally different SCSI controllers. The IC may provide both PCI and SCSI functionality. In this example only the SCSI functionality is represented.
- Typically the chip is physically on a card. However, the ChipOnCard relationship is not depicted in this example.
- The chip realizes the 2 separate SCSI initiator functionalities. These relationships are identified via the **Realizes** associations between the chip and the SCSIControllers.

- **PhysicalMedia (Physical) –** Each of the 4 PhysicalMedia represents the media (platters) of a SCSI hard drive.
- The PhysicalMedia realizes the storage functionality of the drive via the **RealizesPExtent** association.
- The relationship between the drive and its' PhysicalMedia is represented via the **PackagedComponent** association.
- NOTE: The optimizations of having the PhysicalPackage realize the PhysicalExtent via the RealizesPExtent association is permitted. If this optimization were used, then the PhysicalMedia would not be instantiated.

- **PhysicalPackage (Physical) –** Each of the 4 PhysicalPackage represents a SCSI hard drives.
- The PhysicalPackage realizes the "player" functionality of the drive via the **Realizes** association.
- The relationship between the drive and its' PhysicalMedia is represented via the **PackagedComponent** association.
- NOTE: The optimizations of having the PhysicalPackage realize the PhysicalExtent via the RealizesPExtent association is permitted. If this optimization were used, then the PhysicalMedia would not be instantiated.

- **SCSIController (Logical) lowest 2 above upper magenta line–** The 2 SCSIControllers represent the SCSI functionality that the IC provides.
- The relationship between the functionality and the physical IC is identified via the **Realizes** association.
- Each SCSIController accesses 2 of the SCSI hard drives. These relationships are represented via the SCSIInterface associations between the SCSIController and DiskDrives.
- The SCSIController provides an interface for the Logical Unit to the operating environment, SCSI cabinet. This relationship is represented via the **SCSIInterface** association between the SCSIController and PhysicalExtent.

- **DiskDrive (Logical) –** The 4 DiskDrives represent the "player" functionality of the 4 SCSI hard drives.
- The PhysicalPackage realizes the "player" functionality of the drive via the **Realizes** association.
- The relationship between the DiskDrive and the SCSIController is represented via the **SCSIInterface** association.
- The relationship between the "player" and the media is represented via the **MediaPresent** association.

- **PhysicalExtent –** Each of the 4 PhysicalExtents represents the data store functionality of a SCSI logical unit.
- The PhysicalMedia realizes the data store functionality of the drive via the **RealizesPExtent** association.

- The relationship between the "player", DiskDrive, and the media, PhysicalExtent, is represented via the **MediaPresent** association.
- The SCSIController provides an interface for the Logical Unit to the operating environment, SCSI cabinet. This relationship is represented via the **SCSIInterface** association between the SCSIController and PhysicalExtent.
- The PhysicalExtents are grouped into 2 sets of mirrored devices. The aggregate association, PExtentRedundancyComponent, between the PhysicalExtents and the StorageRedundancyGroup, represents this.
- The ProtectedSpaceExtent represents the result of the redundancy. This ProtectedSpaceExtent has a relationship with the underlying PhysicalExtents via the **PSExtentBasedOnPExtent** association.

- **StorageRedundancyGroup (Logical) –** The 2 StorageRedundancyGroups represent a set of mirrored devices.
- The 4 PhysicalExtents are grouped into 2 sets of mirrored devices. The aggregate association, PExtentRedundancyComponent, between the PhysicalExtents and the StorageRedundancyGroups, represents this.

- **ProtectedSpaceExtent (Logical) –** Each ProtectedSpaceExtent represent a mirrored device.
- The property field IsBasedOnUnderlyingRedundancy would indicate that this ProtectedSpaceExtent is the result of a redundancy (represented by the lower magenta (dash dot) box).
- The ProtectedSpaceExtent has a relationship with the 1 of the underlying PhysicalExtents. This relationship is represented via the **PSExtentBasedOnPExtent** association. Since a ProtectedSpaceExtent does not contain any metadata only 1 of the underlying PhysicalExtents has a relationship with the ProtectedSpaceExtent. One PhyscialExtent would contain user data and the other would be the metadata.
- The 2 ProtectedSpaceExtents are striped together. This is represented by the association, **VolumeSetBasedOnPSExtent**

- **VolumeSet (Logical) –** The VolumeSet represents the user data store functionality of a SCSI logical unit.
- The 2 ProtectedSpaceExtents are striped. The association, VolumeSetBasedOnPSExtent, between the VolumeSet and the ProtectedSpaceExtent, represents this.
- One or more DiskPartitions may be placed on the data store. The relationship between the VolumeSet and DiskPartition is represented via the **PartitionBasedOnVolume** association. In this example only one DiskPartition is placed on the data store.
- NOTE: In a mainframe environment there might not be a DiskPartition. In this case, DiskPartition would not be instantiated.
- Access to the Logical Unit, VolumeSet, via the operating environment, host, is represented via the **SCSIInterface** association between the SCSIController and VolumeSet.

- **SCSIController (Logical) middle 1 below blue line–** The SCSIController represents the SCSI functionality of the RAID cabinet.
- The SCSIController provides an interface between the RAID cabinet and the operating environment. This relationship is represented via the **SCSIInterface** association between the SCSIController in the RAID cabinet and the SCSIController on the host.

- **SCSIController (Logical) highest 1 above blue line–** The SCSIController represents the SCSI functionality that exists between the operating environment and the RAID cabinet. Typically, physical objects as described in the SCSI Hard Drive Example would realize it.
- The SCSIController provides an interface between the operating environment and the RAID cabinet. This relationship is represented via the **SCSIInterface** association between the SCSIController on the host and the SCSIController in the RAID cabinet.
- Access to the Logical Unit, VolumeSet, via the operating environment, host, is represented via the **SCSIInterface** association between the SCSIController and VolumeSet.

- **DiskPartition (Logical) –** One or more DiskPartitions may be placed on the data store via a FDISK type operation. In this example only one DiskPartition is placed on the RAID cabinet. If multiple DiskPartitions

were placed on the RAID cabinet, then multiple instances of DiskPartition and its' associations would be instantiated.
- NOTE: In a mainframe environment there might not be a DiskPartition. In that case DiskPartition would not be instantiated.
- The relationship between the VolumeSet and DiskPartition is represented via the **DiskPartitionBasedOnVolumeSet** association.
- A LogicalDisk is placed on the DiskPartition. The relationship between the DiskPartition and the LogicalDisk is represented via the **LogicalDiskBasedOnPartition** association.

- **LogicalDisk (Logical) –** A file system gives a LogicalDisk an identifiable name. For example in a Windows environment, the name is the drive letter, in a Unix environment, the name is the access path, and in a NetWare environment the name is the volume name. A LogicalDisk can only be based on one DiskPartition.
- The relationship between the DiskPartition and the LogicalDisk is represented via the **LogicalDiskBasedOnPartition** association.
- NOTE: In a mainframe environment there might not be a DiskPartition. In that case LogicalDisk would have a LogicalDiskBasedOnVolumeSet relationship with the VolumeSet.
- The relationship between the LogicalDisk and FileSystem is represented via the **ResidesOnExtent** association.

This example uses PhysicalExtents, ProtectedSpaceExtents, and VolumeSet. It could be simplified using StorageExtents and StorageVolume.

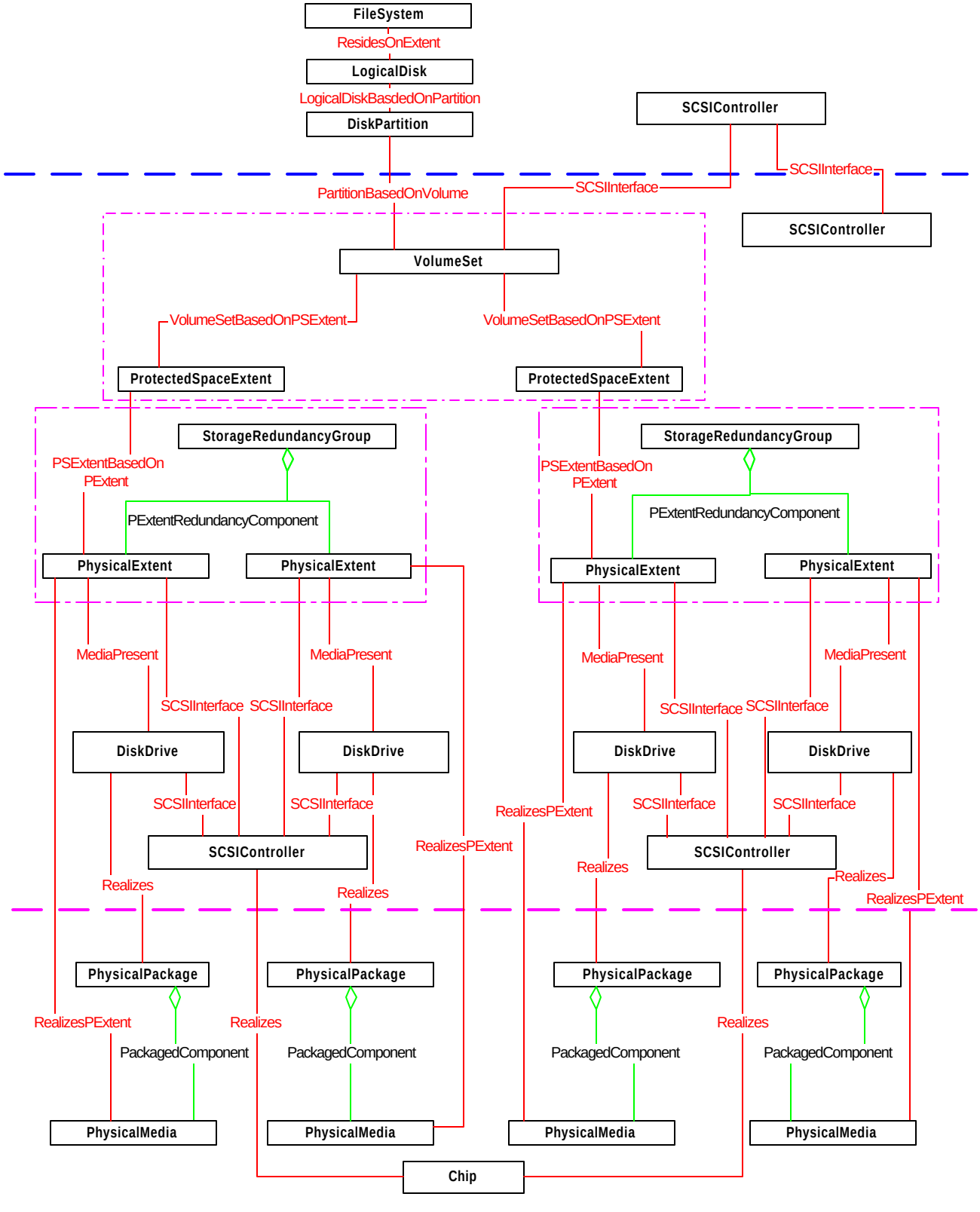

Figure 7-2: SCSI RAID-1 + 0 Cabinet

7.3. Software RAID-1+0 on SCSI Hard Disk Drives

This is example depicts software RAID. The software RAID consists of four hard drives in a RAID-1 + 0 configuration. In this example, all optimizations previously discussed are shown. The objects between the bottom two magenta dashed lines are the physical objects. The remaining objects are logical objects. The objects in the two lower magenta boxes (dash dot) represent the mirroring of data (RAID 1). The objects in the upper magenta box (dash dash dot) represent the striping of data (RAID 0). All physical objects are from the physical model. All logical objects, expect FileSystem, are from the device model. The FileSystem object is from the system model and is not described. The associations are described with the objects they pertain to.

7.3.1. Discussion of Figure 7-3: Software RAID-1 plus striping

- **Chip (Physical)–** Each of the Chips represents the physical integrated circuit (IC). The IC may provide both PCI and SCSI functionality. In this example only the SCSI functionality is represented.
- Typically the chip is physically on a card. However, the ChipOnCard relationship is not depicted in this example.
- The chip realizes the SCSI initiator functionality. These relationships are identified via the **Realizes** associations between the chip and the SCSIController.

- **PhysicalPackage (Physical) –** Each of the PhysicalPackages represents the SCSI hard drive.
- The PhysicalPackage realizes the "player" functionality of the drive via the **Realizes** association.
- The PhysicalPackage realizes the storage functionality of the drive via the **RealizesDiskPartition** association.

- **SCSIController (Logical)–** Each of the SCSIControllers represents the SCSI functionality that the IC provides.
- The relationship between the functionality and the physical IC is identified via the **Realizes** association.
- The SCSIController accesses the SCSI hard drive. This relationship is represented via the **SCSIInterface** association between the SCSIController and DiskDrive.
- The SCSIController provides an interface for the Logical Unit to the operating environment, host. This relationship is represented via the **SCSIInterface** association between the SCSIController and DiskPartition.

- **DiskDrive (Logical) –** Each of the DiskDrives represents the "player" functionality of the SCSI hard drives.
- The PhysicalPackage realizes the "player" functionality of the drive via the **Realizes** association.
- The SCSI hard drive is accessed by the SCSIController. This relationship is represented via the **SCSIInterface** association between the DiskDrive and SCSIController.
- The relationship between the "player" and the data store functionality is represented via the **MediaPresent** association between the DiskDrive and the DiskPartition.

- **DiskPartition (Logical) –** Each of the DiskPartititons represents the data store functionality of a SCSI logical unit.
- The PhysicalPackage realizes the data store functionality of the drive via the **RealizesDiskPartition** association.
- The relationship between the "player", DiskDrive, and the data store, DiskPartition is represented via the **MediaPresent** association.
- The SCSIController provides an interface for the Logical Unit to the operating environment, SCSI cabinet. This relationship is represented via the **SCSIInterface** association between the SCSIController and DiskPartition.
- The DiskPartitions are grouped into 2 sets of mirrored devices. The aggregate association, ExtentRedundancyComponent, between the DiskPartitions and the lower StorageRedundancyGroups, represents this.
- The StorageExtent represents the result of the redundancy. This StorageExtent is **BasedOn** the DiskPartitions and is logically equivalent to the objects in each of the lower magenta boxes (dash dot) beneath it.

- **StorageRedundancyGroup (Logical) lower 2 in dash dot box –** Each of the StorageRedundancyGroups represents a set of mirrored devices.

- The 4 DiskPartitions are grouped into 2 sets of mirrored devices. The aggregate association, ExtentRedundancyComponent, between the DiskPartitions and the lower StorageRedundancyGroups, represents this.

- **StorageExtent (Logical) –** Each of the StorageExtents represents a set of mirrored devices.
- The property field IsBasedOnUnderlyingRedundancy would indicate that this StorageExtent is the result of a redundancy (represented by the lower magenta (dash dot) box).
- The StorageExtent is **BasedOn** 2 underlying DiskPartitions.
- The StorageExtents are striped together. The aggregate association, ExtentRedundancyComponent, between the StorageExtents and the higher StorageRedundancyGroup, represents this.
- The StorageVolume represents the result of the redundancy. The StorageVolume is **BasedOn** the StorageExtents and is logically equivalent to redundancy (the dash dot dot box) beneath it.

- **StorageRedundancyGroup (Logical) higher 1 in dash dot dot box –** The StorageRedundancyGroup represents the striping of the data across the 2 StorageExtents, the 2 sets of mirrored devices.
- The 2 StorageExtents are striped. The aggregate association, ExtentRedundancyComponent, between the StorageExtents and the higher StorageRedundancyGroup, represents this.

- **StorageVolume (Logical) –** The StorageVolume represents the data store functionality of the Software RAID-1 + 0.
- The property field IsBasedOnUnderlyingRedundancy would indicate that this StorageVolume is the result of a redundancy (represented by the upper magenta (dash dot dot) box).
- The StorageVolume is **BasedOn** the underlying StorageExtents.
- A LogicalDisk may be placed on the StorageVolume. The relationship between the StorageVolume and LogicalDisk is represented via the **LogicalDiskBasedOnVolume** association.

- **LogicalDisk (Logical) –** A file system gives a LogicalDisk an identifiable name. For example in a Windows environment the name is the drive letter, in a Unix environment the name is the access path, and in a NetWare environment the name is the volume name. A LogicalDisk can only be based on one StorageVolume or DiskPartition.
- The relationship between the StorageVolume and the LogicalDisk is represented via the **LogicalDiskBasedOnVolume** association.
- The relationship between the LogicalDisk and FileSystem is represented via the **ResidesOnExtent** association. A FileSystem may span multiple LogicalDisks.

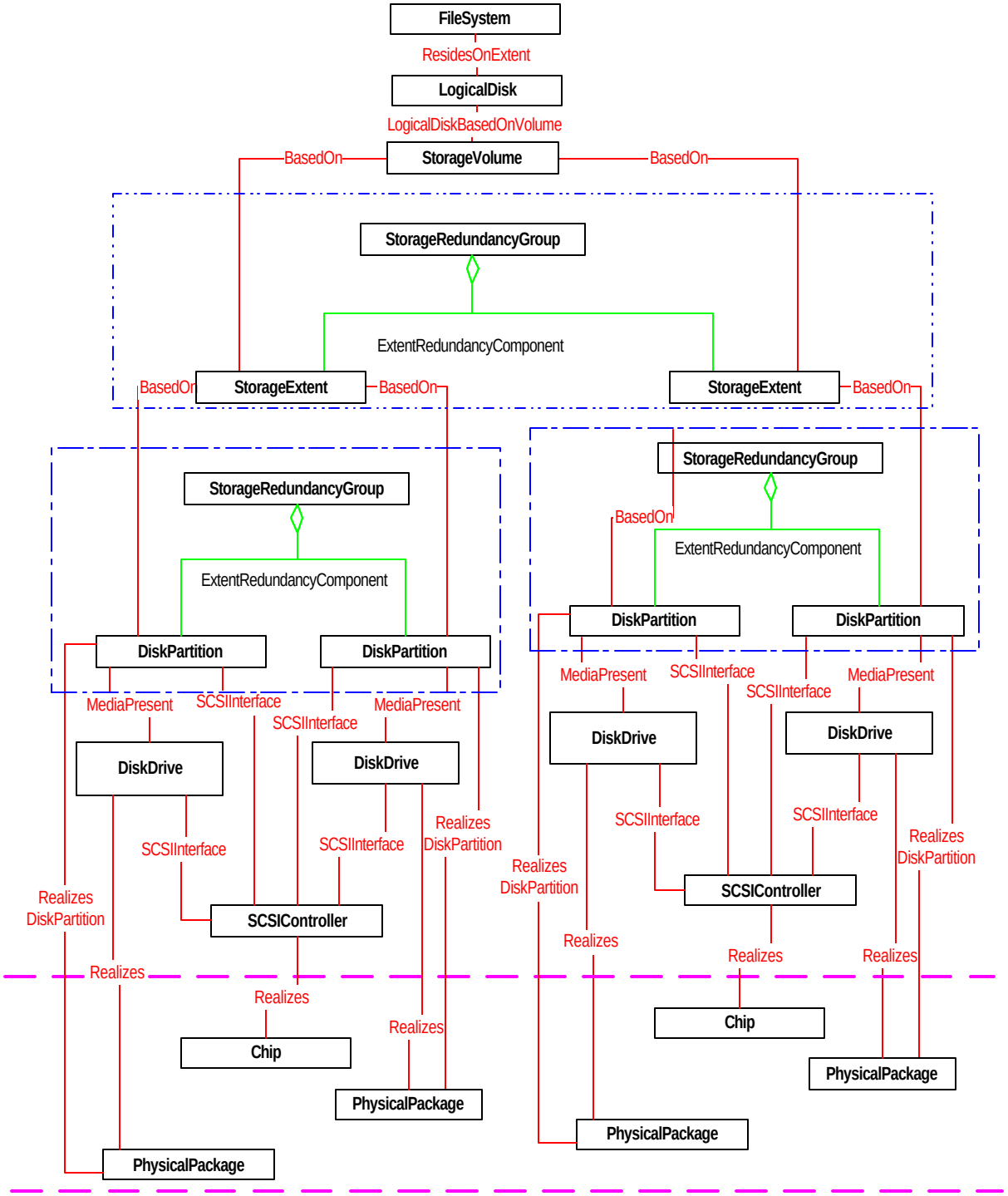

Figure 7-3: Software RAID-1 plus stripi ng

7.4. Snapshot Scenarios

The following depict examples of the various types of snapshots or point in time copies represented by the Snapshot object. For all of the examples Figure 9-4 represents the objects that are initiated before the snapshot or point in time copy occurs.

The **StorageExtent** represents the data store functionality of the media.
- The StorageVolume represents the storage exported to the operating environment. The relationship between the StorageExtent and the StorageVolume is represented via the **BasedOn** association.

The **StorageVolume** represents the data store functionality of the media that is presented to the operating environment.
- The StorageVolume represents the storage exported to the operating environment. The relationship between the StorageExtent and the StorageVolume is represented via the **BasedOn** association.

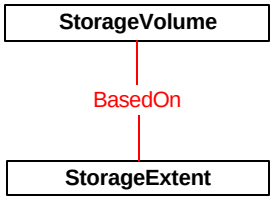

Figure 7-4 - Before Snapshot StorageExtents

7.4.1. Full Copy Snapshot

For the Full Copy case, a copy of the original StorageExtent is made. Figure 9-4 represents the before copy instances, and Figure 9-5 represent the after copy instances.

If the "Full Copy" object is to be treated as a StorageVolume or more general Extent, then it should be modeled as such (i.e., not as an instance of Snapshot but as an instance of a generic StorageExtent or StorageVolume). A new association in CIM V2.3 (CIM_Synchronization) is used to describe the fact that one object is synchronized with another. This association defines a generic model describing that one object (an Extent, a FileSystem, a LogicalFile, etc.) is synchronized with another. It does not address the underlying mechanisms to accomplish this synchronization. This approach solves the problem of instance deletion/recreation when a snapshot/copy is broken. Only the instance of CIM_Synchronization has to be deleted – not the referenced objects.

The **StorageExtent** contained within the magenta box represents the original StorageExtent.

The **StorageVolume** contained within the magenta box represents the original StorageVolume.

The StorageVolume represents the storage exported to the operating environment. The relationship between the StorageExtent and the StorageVolume is represented via the **BasedOn** association.

The **Snapshot** represents the copy of the StorageExtent.

The **SnapshotOfExtent** association between the StorageExtent and the Snapshot describes the relationship between the copy and the original Extent.

The **StorageExtent** right of the Magenta box represents the StorageExtent that contains the copy.

The **BasedOn** association between the StorageExtent right of the magenta box and the Snapshot describes the area used to store the copied data.

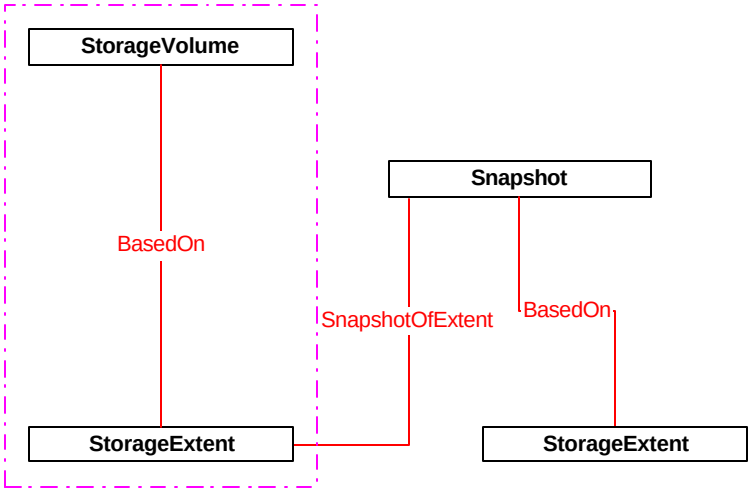

Figure 7-2 Full Copy Snapshot

7.4.2. Before Image Delta Snapshot

For the Before Image Delta case, a point in time copy of the original StorageExtent is made. Figure 9-4 represents the before copy instances, and Figure 9-6 represent the after copy instances.

The **StorageVolume** contained within the magenta box represents the original StorageVolume. The StorageVolume represents the storage exported to the operating environment. If a write is allowed to the original StorageVolume, then the data store represented by the original StorageExtent is updated. If this is the first time these particular blocks of the original StorageExtent is updated, then these original blocks are updated in the data store represented by the Snapshot.

The **StorageVolume** right of the magenta box represents the copied StorageVolume. The StorageVolume represents the storage exported to the operating environment. If a write is allowed to the copied StorageVolume, then the data store represented by the Snapshot is updated.

The **StorageExtent** contained within the magenta box represents the original data store.

The **StorageExtent** right of the magenta box represents the underlying data store of the snapshot.

The **Snapshot** represents the data store for the changes that are required to maintain the data store of the copied StorageVolume.

The original StorageVolume (within the magenta box) represents the storage exported to the operating environment. The **BasedOn** association represents the relationship between this StorageVolume and its data store, the original StorageExtent (within the magenta box).

The **SnapshotOfExtent** association represents the relationship between the original StorageExtent (within the magenta box) and the Snapshot.

The relationship between the Snapshot and its data store, StorageExtent (right of the magenta box) is represented via the **BasedOn** association.

The copied StorageVolume (right of the magenta box) represents the storage exported to the operating environment. The copied StorageVolume is based upon two different data stores. Each data store has a **BasedOn** relationship to the copied StorageVolume. The BasedOn association to the original StorageExtent (within the magenta box) represents the relationship to the data store that contains unchanged data blocks. The BasedOn association to the Snapshot represents the relationship to the data store that contains the changed data blocks.

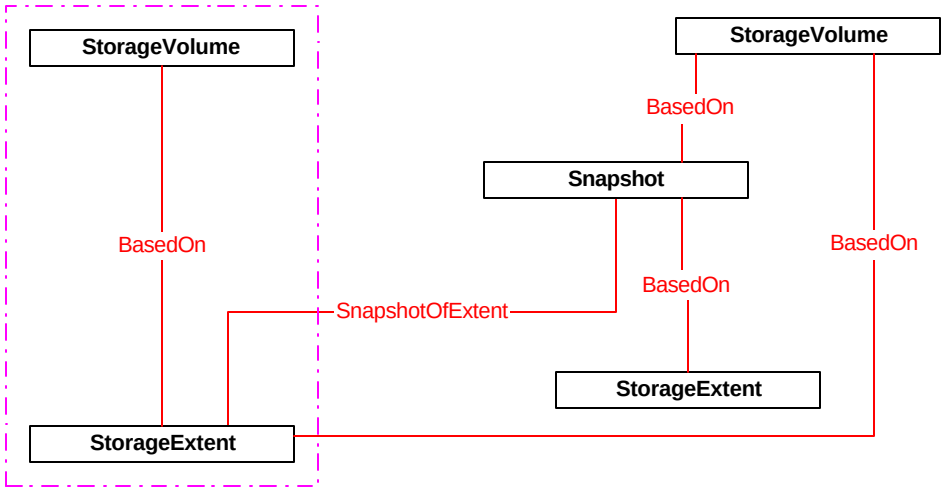

Figure 7-3 Before Image Delta Snapshot

7.4.3. After Image Delta Snapshot

For the After Image Delta case, a point in time copy of the original StorageExtent is made. Figure 9-4 represents the before copy instances, and Figure 9-7 represent the after copy instances.

The **StorageVolume** contained within the magenta box represents the original StorageVolume. The StorageVolume represents the storage exported to the operating environment. If a write is allowed to the original StorageVolume, then the data store represented by the snapshot is updated.

The **StorageVolume** right of the magenta box represents the copied StorageVolume. The StorageVolume represents the storage exported to the operating environment. If a write is allowed, then additional objects should be instantiated to reflect the implementation.

The **StorageExtent** contained within the magenta box represents the original StorageExtent.

The **StorageExtent** left of the magenta box represents the underlying data store of the snapshot.

The **Snapshot** represents the data store for the changes that are required to maintain the data store of the original StorageVolume.

The original StorageVolume (within the magenta box) represents the storage exported to the operating environment. The original StorageVolume is based upon two different data stores. . Each data store has a **BasedOn** relationship to the original StorageVolume. The BasedOn association to the original StorageExtent (within the magenta box) represents the relationship to the data store that contains unchanged data blocks. The BasedOn association to the Snapshot represents the relationship to the data store that contains the changed data blocks.

The **SnapshotOfExtent** association represents the relationship between the original StorageExtent (within the magenta box) and the Snapshot.

The relationship between the Snapshot and its data store, StorageExtent (left of the magenta box) is represented via the **BasedOn** association.

The **BasedOn** association represents the relationship between this copied StorageVolume and its data store, the original StorageExt ent (within the magenta box).

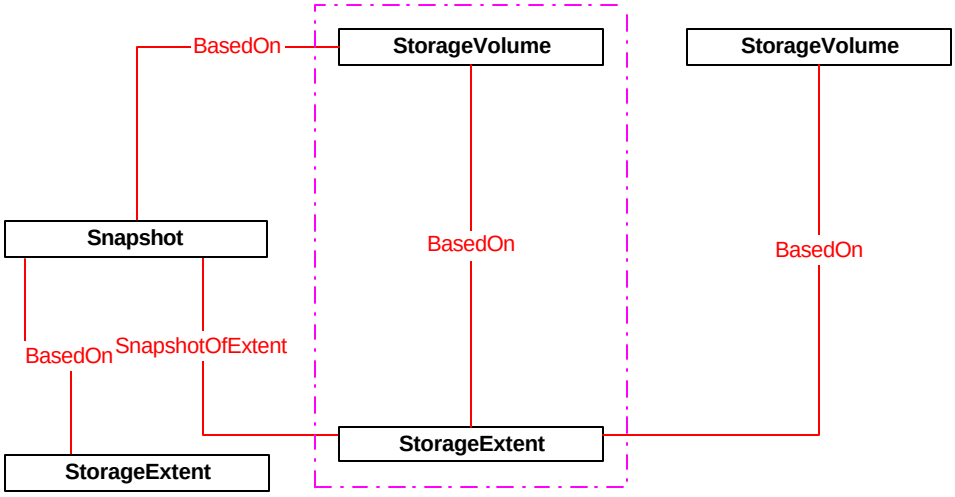

Figure 7-4 After Image Delta Snapshot

8. Mappings

Almost the entire standard DMI groups and attributes related to storage can be mapped into the CIM storage model. Mappings between DMI and CIM are discussed in section 8.1.

Since there are no standard SNMP MIB for storage, no mapping for SNMP is discussed.

Several specialization were derived from definitions in "SCSI Controller Commands - 2 (SCC-2)", developed by the National Committee for Information Technology Standards, ANSI/NCITS 318-1998, published in 1998. Mappings between SCC-2 and CIM are discusses in section 8.2.

8.1. DMI to CIM

The follow table reflects the mapping strings defined by the CIM Schema V2.2 of the storage related Wired for Management (WfM) required and recommended DMI Groups.

* Indicates that the attribute is not directly mapped in the CIM Schema but could be mapped by appropriately defining properties in an object's instantiation or in an extension schema.
+ Indicates that the attribute has been identified as an area of work to be completed in the CIM Schema V2.3.

All CIM object names should be preceded with a "CIM_" to be valid. This was omitted throughout this table for brevity.

DMI Group / Attribute Name	CIM Property
Aggregate Physical Extent|001	
Aggregate Physical Extent Index	AggregatePExtent uniquely identified by its key properties
Number Of Blocks	AggregatePExtent.NumberOfBlocks
Number Of Blocks Of Check Data	AggregatePExtent.BlocksOfCheckData
Aggregate Protected Space Extent|001	
Aggregate Protected Space Extent Index	AggregatePSExtent uniquely identified by its key properties
Number Of Blocks	AggregatePSExtent.NumberOfBlocks
Bus Port|002	
Bus Port Index	Bus Ports are modeled as Controllers, NetworkAdapters and PhysicalConnectors from the logical and physical CIM aspects. All are uniquely identified by their key properties. Both Controllers and NetworkAdapters are "Realize"d in PhysicalElements, including their PhysicalConnectors.
Protocol	Controller.ProtocolSupported, or conveyed as a subclass of NetworkAdapter
ProtocolDescription	NetworkAdapter.Description or Controller.Description. For Controller, this info is better stored in an explicit property, ProtocolDescription, in the Controller class. Network-oriented protocol details are described as subclasses of ProtocolEndpoint (in the Networks Model).
Signal Characteristics	Defined for a subclass of Controller, SCSIController.SignalCapabilities. It describes the Controller's ability to support various signal characteristics. The SCSIInterface Controller's ability to support various signal characteristics. The SCSIInterface association Controller's ability to support various signal characteristics. The SCSIInterface association describes the signalling currently in use between a SCSIController and its downstream device(s) - using the SCSISignal property.
Address Descriptor	SCSI address data is included as properties of the SCSIInterface association. NetworkAdapter has a PermanentAddress property. Addressing data could be used as the Controller's or NetworkAdapter's DeviceID or stored in the OtherIdentifyingInfo string array, inherited from LogicalDevice.

Isochronous	
Maximum Width	SCSIController.MaxDataWidth
Maximum Transfer Rate	NetworkAdapter.MaxSpeed, SCSIController.MaxTransferRate, SerialController.MaxBaudRate
Maximum Number Of Attachments	Controller.MaxNumberControlled
+Connector Type	PhysicalConnector.ConnectorType ("SCSI VHDCI shielded (68 pins)" should be added to the enumeration)
+Connector Type Description	PhysicalConnector.Description (Better stored in an explicit property, OtherTypeDescription, on PhysicalConnector)
Connector Gender	PhysicalConnector.ConnectorType includes "Male" and "Female" in the enumeration.
Bus Port Association|001	
Bus Port Association Index	The association of bus ports is described using the DeviceConnection association, or its subclass, ControlledBy, when a Controller instance is involved. Associations are uniquely identified by their key properties.
Negotiated Speed	DeviceConnection.NegotiatedSpeed
Negotiated Width	DeviceConnection.NegotiatedDataWidth
Component Spare Association|001	
Component Spare Index	SpareGroup and ExtraCapacityGroup are subclasses of RedundancyGroup (which defines instances' keys). Sparing includes failover concepts, while Extra Capacity Groups can be used to describe load balancing. Membership in a Spare or ExtraCapacityGroup is indicated by an instance of RedundancyComponent. The entities that "do" the sparing are indicated using the ActsAsSpare association.
Spare Functioning State	Active versus Inactive sparing is described by the HotStandby boolean property of the ActsAsSpare association. Load balancing is described by membership in an ExtraCapacityGroup with the LoadBalancedGroup boolean property set to TRUE.
Disks Mapping Table|001	
Storage Type	PhysicalMedia.MediaType + Instantiation of the appropriate MediaAccessDevice subclass (to indicate the type of media reader). The StorageExtents currently accessed by the MediaAccessDevice are identified by following the MediaPresent association from the Device to one or more StorageExtents.
Disk Index	A Disk is an instantiation of a StorageExtent or one of its subclasses. It is related to a Partition as described for Partition Index, directly below. The drive mechanism is defined as a DiskDrive object, a subclass of MediaAccessDevice.
Partition Index	Partitions can be built on lower level StorageExtent(s). This is described using the BasedOn relationship between Storage Extents. Alternately, Partitions may be directly "realized" on PhysicalMedia. This relationship is indicated by the RealizesDiskPartition association, defined in the Physical Model.
Disks|003	
Storage Type	PhysicalMedia.MediaType + Instantiation of the appropriate MediaAccessDevice subclass (to indicate the type of media reader). The StorageExtents currently accessed by the MediaAccessDevice are identified by following the MediaPresent association from the Device to one or more StorageExtents.
Disk Index	Device uniquely identified by its key properties
Storage Interface Type	PhysicalConnector.ConnectorType for the Connector of the Disk's PhysicalPackage. This is found by traversing the ConnectorOnPackage association. Additional info is provided by following the ControlledBy association from the MediaAccessDevice to the Controller object that manages it. The Controller.ProtocolSupported property contains the info on the low level protocol.

Storage Type	PhysicalMedia.MediaType + Instantiation of the appropriate MediaAccessDevice subclass (to indicate the type of media reader). The StorageExtents currently accessed by the MediaAccessDevice are identified by following the MediaPresent association from the Device to one or more StorageExtents.
Disk Index	Device uniquely identified by its key properties
Storage Interface Type	PhysicalConnector.ConnectorType for the Connector of the Disk's PhysicalPackage. This is found by traversing the ConnectorOnPackage association. Additional info is provided by following the ControlledBy association from the MediaAccessDevice to the Controller object that manages it. The Controller.ProtocolSupported property contains the info on the low level protocol.
Interface Description	Inherited from ManagedSystemElement, the Description property.
Media Loaded	Information provided by following the MediaPresent association between the MediaAccessDevice and StorageExtents accessed through the Device.
Removable Drive	Information found in Physical Package.Removable for the Package which "realizes" The appropriate StorageExtent.
Removable Media	MediaAccessDevice.Capabilities, PhysicalMedia.MediaType
DeviceID	SCSIInterface.TargetID
Logical Unit Number	SCSIInterface.TargetLUN
*Number of Physical Cylinders	
*Number of Phys Sectors/Track	
*Number of Physical Heads	
*Phys Cylinder for Write Precompensation	
*Physical Cylinder for Landing Zone	
*Sector Size	(Not typically instrumented) Could instantiate a StorageExtent or one of its subclasses (such as PhysicalExtent or AggregatePExtent) using "sector size" as the StorageExtent.BlockSize.
Total Physical Size	PhysicalMedia.Capacity or calculated by multiplying StorageExtent.BlockSize by StorageExtent.NumberOfBlocks.
Number of Current Bad Blocks or Sectors	Obtained by totaling all StorageError objects associated with the StorageExtent (discovered by following the StorageDefect relationship).
Partitions	Partitions can be built on lower level StorageExtent(s). This is described using the BasedOn relationship between Storage Extents. Alternately, Partitions may be directly "realized" on PhysicalMedia. This relationship is indicated by the RealizesDiskPartition association. To obtain the total number of Partitions, one must follow the appropriate associations (SystemDevice) from ComputerSystem to its component LogicalDevices and total the number of Partitions found.
Physical Location	Indicated by instantiation of Location object(s) associated with a PhysicalMedia object or with a PhysicalPackage that "realizes" the MediaAccessDevice for this Disk (both PhysicalMedia and PhysicalPackage are subclasses of PhysicalElement). A Storage Extent is associated with a PhysicalMedia object using the Realizes relationship or one of its subclasses (for example, RealizesPExtent). Therefore, one can follow the Realizes relationship from the StorageExtent to the PhysicalElement and then follow the PhysicalElement's associations to the Location object. As an optimization, the location data might be found in the Container association, the LocationWithinContainer string property.
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Disk's physical "realization". Alternately, the PhysicalElement(s) that realize The Disk could be associated with other Elements that are to be replaced as a "set", using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object
Security Settings	MediaAccessDevice.Security

Fibre Channel Bus Port Extensions|001	
Bus Port Index	FibrePort uniquely identified by its key properties
End to End Credit	FibrePorts participate in active log-ins with other Ports. This is the only time that credits have meaning. This info is stored in the OriginatorEndCredit and ResponderEndCredit properties of the FibrePortActiveLogin association. More than 1 login can be active at a time.
Buffer to Buffer Credit	FibrePorts participate in active log-ins with other Ports. This is the only time that credits have meaning. This info is stored in the OriginatorBufferCredit and ResponderBufferCredit properties of the FibrePortActiveLogin association. More than 1 login can be active at a time.
*Link Type	Some of this information is located in the PhysicalLink object , in the MediaType property. It describes the connecting wiring/media attached to the Port. (Some but not all data is mapped.)
Flow Control Class Type	FibrePortActiveLogin.NegotiatedCOS ("Class-1/Class-2" appears to represent Intermix, where Class 2 and 3 frames can be used over a Class 1 connection. In CIM, this is viewed as a special case of Class 1 Service.)
Flow Control Ack Type	FibrePortActiveLogin.AckModel
*Fabric Topology	(Old terminology from FC-PH. To be addressed in future versions of the CIM Schema, by the definition of an FC "network".)
Mass Storage Association|001	
Association Index	Associations uniquely identified by their key properties
Type	Physical Organization described by the Container association and its subclasses (from the Physical Model). Logical Organization described by the BasedOn association and its subclasses (from the Device Model). Logical to Physical Organization described by the Realizes association and its subclasses (defined in the Core and Physical Models). Redundancy Organization described by the RedundancyComponent association and its subclasses (defined in the Device Model). Spare Organization described by the RedundancyComponent and ActsAsSpare associations (in the Device Model). Cache is described as AssociatedMemory (from the Device Model). FRU is associated with its component PhysicalElements and software using the FRUPhysicalElements and FRUIncludesSoftwareFeature associations (from the Core and Application Models). Operational State information is not associated but included in the LogicalDevice and DeviceErrorCounts objects. Statistical data is related to mass storage using the ErrorCountersForDevice or DeviceStatistics associations (from the Device and Core Models). SubcomponentSoftware is described using the DeviceSoftware association (from the Device Model). Worldwide Identifier is not associated but defines the object's key (ie, DeviceID), or is stored in the OtherIdentifyingInfo string array property, inherited from LogicalDevice. OtherIdentifyingInfo string array property, inherited from LogicalDevice.
Reference 1	Association's reference
Reference 2	Association's reference
Mass Storage Statistics|001	
Mass Storage Statistics Index	A subclass of DeviceStatisticalInformation. (Instances are uniquely identified by the key structure of DeviceStatisticalInformation.) Stat info is associated with a "mass storage" device using an instance of DeviceStatistics.
Read Commands Sent	
Write Commands Sent	
Read Commands Received	
Write Commands Received	
Data Unit	
I/O Range 1 Sent	

I/O Range 2 Sent	
I/O Range 3 Sent	
I/O Range 4 Sent	
I/O Range 1 Received	
I/O Range 2 Received	
I/O Range 3 Received	
I/O Range 4 Received	
Fault Time	
Fault Code	Devices report error information using the LastErrorCode property, inherited from LogicalDevice.
Number Of Errors	More granular error reporting is possible using the DeviceErrorCounts statistical class.
Number Of Retries	SCSIInterface.SCSIRetries
Parallel Port Index	ParallelController uniquely identified by its key properties
Parallel Base I/O Address	MemoryMappedIO.StartingAddress - The MemoryMappedIO object is found by following the AllocatedResource relationships from the ParallelController instance.
IRQ Used	IRQ.IRQNumber - The IRQ object is found by following the AllocatedResource relationships from the ParallelController instance.
Logical Name	Represents the LogicalDevice connected to this parallel port. Found by traversing the ControlledBy association from the ParallelController instance.
Connector Type	Must follow the Realizes relationship from the ParallelController to its PhysicalElement and then query its associated PhysicalConnectors.
Connector Pinout	PhysicalConnector.ConnectorPinout - The appropriate PhysicalConnector is found by following the Realizes relationships from the ParallelController instance.
DMA Support	ParallelController.DMASupport
Parallel Port Capabilities	ParallelController.Capabilities
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
Parallel Port Security Settings	ParallelController.Security
Partition|002	
Partition Index	DiskPartition uniquely identified using its key properties
Partition Name	Inherited from LogicalDevice to Partition, the DeviceID property
Partition Size	Inherited from StorageExtent, the product of BlockSize and NumberOfBlocks
Free Space	FileSystem.AvailableSpace - File Systems have a Dependency relationship (ResidesOnExtent) to StorageExtent. Extents do not have "free space" - File Systems do.
Partition Label	The DeviceID of the LogicalDisk that is BasedOn the Partition (via the LogicalDiskBasedOnPartition association)
File System	As above, File Systems have a Dependency relationship – ResidesOnExtent - to StorageExtents.
Compressed	FileSystem.CompressionMethod
Encrypted	FileSystem.EncryptionMethod
Number Of Disks Occupied	DiskPartitions can not span PhysicalMedia unless there is some intervening redundancy (for example, RAID mirroring or striping). In this case, DiskPartitions are built on a lower level StorageExtent using the BasedOn relationship. The "typical" case is where a Partition is BasedOn a StorageVolume. The explicit association, DiskPartitionBasedOnVolume, defines this. In turn, the StorageVolume is BasedOn underlying StorageExtents that are ultimately Realized in PhysicalMedia. To obtain the number of disks occupied, follow these associations and total the number of PhysicalMedia found. If the Partition is directly Realized on a single PhysicalMedia, this is indicated by the RealizesDiskPartition association, defined in the Physical Model.
Physical Extent|001	
Physical Extent Index	PhysicalExtent uniquely identified by its key properties StartingAddress is a

	property of the association.
Number Of Blocks	NumberOfBlocks, inherited from StorageExtent
Block Size	BlockSize, inherited from StorageExtent
Granularity Unit	N/A as CIM properties have a Units qualifier which can be overridden.
Start Check Data Interleave	PhysicalExtent.UnitsBeforeCheckDataInterleave
Units Of Check Data	PhysicalExtent.UnitsOfCheckData
Units Of User Data	PhysicalExtent.UnitsOfUserData
Physical Memory Array|001 - Can be VolatileStorage, NonVolatileStorage or CacheMemory	
Memory Array Table Index	Memory and memory groupings uniquely defined by their key properties
Memory Array Location	Information available by traversing the Realizes relationship from Logical to Physical Element and then following the PhysicalElementLocation association to a Location object. As an optimization, the location data might be found in the Container association, the LocationWithinContainer string property.
Memory Array Use	Information could be placed in the Purpose property, inherited from StorageExtent to all Memory classes. Alternately, this data is available by examining the Memory subclass that is instantiated (Volatile Storage, NonVolatile Storage, Cache Memory) and checking whether this Memory is associated with a Device or not (for example, video memory would be associated with a Video Controller using the AssociatedMemory relationship).
Maximum Memory Capacity	Instance of a MemoryCapacity object, associated with a PhysicalElement (via the ElementCapacity relationship).
Number of Memory Device Sockets	Information can be obtained by enumerating the Memory-related PhysicalConnectors for the HostingBoard or (Memory) Card.
Numb of Mem Dev Sockets Used	Addressed by enumerating the MemoryOnCard association between Memory components and a HostingBoard or (Memory) Card.
Memory Error Correction	Memory.ErrorMethodology, an override of StorageExtent's ErrorMethodology property
Array Error Type	Memory.ErrorInfo
Last Error Update	Memory.ErrorTime
Error Operation	Memory.ErrorAccess
Error Data Size	Memory.ErrorTransferSize
Error Data	Memory.ErrorData
Vendor Syndrome	Memory.AdditionalErrorData
Error Address	Memory.ErrorAddress
Error Resolution	Memory.ErrorResolution
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Memory's physical "realization". Alternately, the PhysicalElement(s) that realize the Memory (instances of PhysicalMemory or Cards with Memory) could be associated with other Elements that are to be replaced as a "set", using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
Protected Space Extent|001	
Protected Space Extent Index	ProtectedSpaceExtent uniquely identified by its key properties
Start Address	StorageExtents are built on other extents as described by the BasedOn association. StartingAddress is a property of the association.
Number Of Blocks	ProtectedSpaceExtent.NumberOfBlocks
Block Size	ProtectedSpaceExtent.BlockSize
User Data Stripe Granularity Units	N/A as CIM properties have a Units qualifier which can be overridden.
User Data Stripe Depth	ProtectedSpaceExtent.UserDataStripeDepth
Redundancy Group|001	
Redundancy Group Index	StorageRedundancyGroup uniquely identified by its key properties

Redundancy Type	StorageRedundancyGroup.TypeOfAlgorithm
Serial Ports|004	
Serial Port Index	SerialController uniquely identified by its key properties
Serial Base I/O Address	MemoryMappedIO.StartingAddress - The MemoryMappedIO object is found by following the AllocatedResource relationships from the SerialController instance.
IRQ Used	IRQ.IRQNumber - The IRQ object is found by following the AllocatedResource relationships from the SerialController instance.
Logical Name	Inherited from LogicalDevice to SerialController, the DeviceID property
Connector Type	Must follow the Realizes relationship from the SerialController to its PhysicalElement and then query its associated PhysicalConnectors.
Maximum Speed	SerialController.MaxBaudRate
Serial Port Capabilities	SerialController.Capabilities
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
Serial Port Security Settings	SerialController.Security
SSA Bus Port Extensions|001	
+Bus Port Index	
+Port Mode	
+Satisfaction Quota	
Storage Controller|001	
Controller Index	The Controller object and its subclasses are uniquely identified by their key properties. SCSIController corresponds most closely to "Storage Controller".
Identification	Hardware data (name, brand, revision level) is conveyed as properties of PhysicalElement and associated with the Controller using a Realizes relationship.
Protection Management	SCSIController.ProtectionManagement
Bus Master	Controllers communicate with downstream devices - as indicated by the ControlledBy association. The AccessState property of ControlledBy defines the Controller as the "active" one for the connection. Since this may vary by connection, the data is in the association and not in the object itself.
Seconds Since Last Power-Up	PowerOnHours, inherited from LogicalDevice (similar information but less granular)
Storage Devices|001	
Storage Device Index	MediaAccessDevice or MediaTransferDevice uniquely identified by its key properties
Type	Subclasses of MediaAccessDevice convey "Type" information for media players. Data on media changers and pickers is found in subclasses of MediaTransferDevice. Regarding the types of media that can be placed in the players/changers, StorageMediaLocations (subclasses of PhysicalPackage) participate in "Realize"ing MediaAccessDevices and MediaTransferDevices. StorageMediaLocations include MediaTypesSupported and MediaSizesSupported as properties.
Type Description	MediaAccessDevice.Description, MediaTransferDevice.Description, StorageMediaLocation.Description
Sub-Identifier	Information such as SCSI LUN is found in the SCSIInterface association. Other identifier data can be placed in the OtherIdentifyingInfo array, inherited from LogicalDevice.
Media Data Block Size	MediaAccessDevice has DefaultBlockSize, MinBlockSize and MaxBlockSize properties. This attribute, however, deals with the media's block size - where media is defined as a StorageExtent (or one of its subclasses) with a BlockSize property. StorageExtents are associated with MediaAccessDevices using the MediaPresent relationship.
Formatted Media Capacity	StorageExtent.BlockSize multiplied by NumberOfBlocks, where a StorageExtent is associated with a MediaAccessDevice using the MediaPresent relationship.

Removable Device	MediaAccess and TranferDevices are realized in PhysicalPackages (described using the Realizes association). PhysicalPackages have a Removable boolean property.
Device Loaded	MediaAccess and TranferDevices are realized in PhysicalPackages (described using the Realizes association). PhysicalPackages may be contained by other Packages using the Container associaton or one of its subclasses.
Removable Media	MediaAccessDevice.Capabilities array (Enumeration includes "Supports Removeable Media")
Media Loaded	MediaPresent association between a MediaAccessDevice and a StorageExtent, or PhysicalMediaInLocation between a PhysicalMedia and a StorageMediaLocation
Compression	MediaAccessDevice.Capabilities array (Enumeration includes Compression") and a CompressionMethod string property can be used to describe the algorithm/mechanism.
Encryption	MediaAccessDevice.Capabilities array (Enumeration includes "Encryption")
SubComponent Software|001	
Software Index	SoftwareElement uniquely identified via its key properties. SoftwareElements are associated to LogicalDevices using the DeviceSoftware relationship.
Type	DeviceSoftware.Purpose
Vendor	SoftwareElement.Manufacturer
Version	SoftwareElement.Version
Description	SoftwareElement.Description, inherited from ManagedSystemElement
Identification Code	SoftwareElement.IdentificationCode
Language Edition	SoftwareElement.LanguageEdition
Interface Description	SoftwareElement.TargetOperatingSystem (String attribute is converted to an enumerated integer.)
*Interface Version	
System Cache|004	
System Cache Index	CacheMemory uniquely identified by its key properties
System Cache Level	CacheMemory.Level
System Cache Speed	PhysicalMemory.Speed
System Cache Size	Inherited from StorageExtent to CacheMemory, cache size is the product of the properties, BlockSize (ie, bytes) and NumberOfBlocks. Alternately, the information is available in PhysicalMemory.Capacity.
System Cache Write Policy	CacheMemory.WritePolicy
System Cache Error Correction	Inherited from StorageExtent to CacheMemory, the ErrorMethodology property
FRU Group Index	Requires instantiation of a FRU object and a FRUPhysicalElements association to the Memory's physical "realization". Alternately, the PhysicalElement(s) that realize the Memory (instances of PhysicalMemory or Cards with Memory) could be associated with other Elements that are to be replaced as a "set", using the ReplacementSet object and the ParticipatesInSet association (defined in the Physical Model).
Operational Group Index	Operational state data is inherited as properties of the LogicalDevice object.
System Cache Type	CacheMemory.CacheType
Line Size	CacheMemory.LineSize
*Volatility	
Replacement Policy	CacheMemory.ReplacementPolicy
Read Policy	CacheMemory.ReadPolicy
Flush Timer	CacheMemory.FlushTimer
Associativity	CacheMemory.Associativity
Volume Set|001	
Volume Set Index	VolumeSet uniquely identified by its key properties
Name	Name, inherited from ManagedSystemElement

Total Storage Capacity	ConsumableBlocks multiplied by BlockSize (properties inherited from StorageExtent) describe total available capacity.
Protected Space Stripe Length	VolumeSet.PSExtentStripeLength
Protect Sp Ext Interleave Depth	VolumeSet.PSExtentInterleaveDepth
WorldWide Identifier|001	
Worldwide Identifier Index	N/A - Worldwide Identifiers define an object's key or are stored in the OtherIdentifyingInfo property for a System or LogicalDevice.
Worldwide Identifier Type	Dictated by the object being instantiated
Worldwide Identifier	System.Name, LogicalDevice.DeviceID or stored in OtherIdentifyingInfo

8.2. SCSI Controller Command (SCC) to CIM

Several specialization were derived from definitions in "SCSI Controller Commands - 2 (SCC-2)", developed by the National Committee for Information Technology Standards, ANSI/NCITS 318-1998, published in 1998

SCSI Controller Command Set Data Fields	**Corresponding CIM Data Fields**
Volume Set Descriptor	
LUN_V	CIM_VolumeSet.DeviceID
GRANULARITY OF UNITS	------Units are defined to be bytes
STATE OF THE VOLUME SET	CIM_VolumeSet.VolumeStatus
PS_EXTENT STRIPE LENGTH	CIM_VolumeSet.PSExtentStripeLength
PS_EXTENT INTERLEAVE DEPTH	CIM_VolumeSet.PSExtentInterleaveDepth
REPORT VOLUME SET DESCRIPTOR LIST LENGTH	CIM_VolumeSetBasedOnPSExtent
Volume Set Protected Space Extent Descriptor	
LUN_P	CIM_PSExtentBasedOnPExtent. Antecedent->DeviceID
START LBA_PS	CIM_VolumeSetBasedOnPSExtentStartingAddress
NUMBER OF LBA_PSs	CIM_VolumeSetBasedOnPSExtent. EndingAddress
NUMBER OF BYTES PER LBA_PS	CIM_ProtectedSpaceExtent.Blocksize
NOCHKSKIP	CIM_VolumeSetBasedOnPSExtent. LBAMappingIncludesCheckData
INCDEC	CIM_VolumeSetBasedOnPSExtent. LBAsMappedByDecrementing
LUN_R	CIM_VolumeSetBasedOnPSExtent => CIM_ProtectedSpaceExtent => CIM_PSExtentBasedOnPExtent => CIM_PextentRedundancyComponent.Dependent->DeviceID
USER DATA STRIPE DEPTH	CIM_ProtectedSpaceExtent. UserDataStripeDepth
Basic Volume Parameters	
EQSPRD	----------Not represented.
STATE OF THE VOLUME SET	----------Not represented
CAPACITY	CIM_AggregatePSExtent.NumberOfBlocks
BYTES PER BLOCK	CIM_AggregatePSExtent.Blocksize
REPORT BASIC VOLUME SET PERIPHERAL DEVICE DESCRIPTOR LIST	CIM_AggregatePSExtentBasedOnAggregatePExtent Or CIM_AggregatePSExtentBasedOnPExtent
Basic Volume Set Peripheral Device Descriptor	
LUN_P	CIM_AggregatePExtent.DeviceID or CIM_PhysicalExtent.DeviceID.
WEIGHTING OF USER DATA	Derive from CIM_AggregatePExtent. NumberOfBlocks and BlocksOfCheckData or from CIM_PhysicalExtent UnitsOfUserData and NumberOfBlocks.
ASSOCIATED REDUNDANCY GROUPS LIST of LUN_Rs	CIM_AggregateRedundancyComponent.GroupComponent->DeviceID