Document Number: DSP1047

Date: 2010-04-22

Version: 1.0.0

Storage Resource Virtualization Profile

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Figures

Tables

Foreword

This profile was prepared by the System Virtualization, Partitioning and Clustering Working Group of the DMTF.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability.

The authors wish to acknowledge the following people.

- Editors:
  - Michael Johanssen – IBM
  - Mike Walker – IBM
- Participants from the DMTF System Virtualization, Partitioning and Clustering Working Group:
  - Gareth Bestor – IBM
  - Jim Fehlig – Novell
  - Michael Johanssen – IBM
  - Larry Lamers – VMware
  - Andreas Maier – IBM
  - John Parchem – Microsoft
  - Shishir Pardikar – XenSource
  - Nihar Shah – Microsoft
  - David Simpson – IBM
  - Mike Walker – IBM

Introduction

The information in this specification should be sufficient for a provider or consumer of this data to identify unambiguously the classes, properties, methods, and values that shall be instantiated and manipulated to represent and manage the components described in this document. The target audience for this specification is implementers who are writing CIM-based providers or consumers of management interfaces that represent the components described in this document.

Document conventions

Typographical conventions

The following typographical conventions are used in this document:

- Document titles are marked in *italics*.

Experimental material

Experimental material has yet to receive sufficient review to satisfy the adoption requirements set forth by the DMTF. Experimental material is included in this document as an aid to implementers who are interested in likely future developments. Experimental material may change as implementation experience is gained. It is likely that experimental material will be included in an upcoming revision of the document. Until that time, experimental material is purely informational.

The following typographical convention indicates experimental material:

EXPERIMENTAL

Experimental material appears here.

EXPERIMENTAL

In places where this typographical convention cannot be used (for example, tables or figures), the "EXPERIMENTAL" label is used alone.

Storage Resource Virtualization Profile

1 Scope

This profile is a component profile that extends the management capabilities of the referencing profile by adding the support to represent and manage the allocation of storage to virtual systems.

2 Normative references

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

DMTF DSP0004, *CIM Infrastructure Specification* 2.3
http://www.dmtf.org/standards/published_documents/DSP0004_2.3.pdf

DMTF DSP0200, *CIM Operations over HTTP* 1.3
http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf

DMTF DSP0207, *WBEM URI Mapping* 1.0
http://www.dmtf.org/standards/published_documents/DSP0207_1.0.pdf

DMTF DSP1001, *Management Profile Specification Usage Guide* 1.0
http://www.dmtf.org/standards/published_documents/DSP1001_1.0.pdf

DMTF DSP1033, *Profile Registration Profile* 1.0
http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf

DMTF DSP1041, *Resource Allocation Profile* 1.1
http://www.dmtf.org/standards/published_documents/DSP1041_1.1.pdf

DMTF DSP1042, *System Virtualization Profile* 1.0
http://www.dmtf.org/standards/published_documents/DSP1042_1.0.pdf

DMTF DSP1043, *Allocation Capabilities Profile* 1.0
http://www.dmtf.org/standards/published_documents/DSP1043_1.0.pdf

DMTF DSP1054, *Indications Profile* 1.0
http://www.dmtf.org/standards/published_documents/DSP1054_1.0.pdf

DMTF DSP1057, *Virtual System Profile* 1.0
http://www.dmtf.org/standards/published_documents/DSP1057_1.0.pdf

IETF RFC1738, *Uniform Resource Locator (URL)*
http://tools.ietf.org/html/rfc1738

IETF RFC3986, *Uniform Resource Identifier (URI): Generic Syntax*
http://tools.ietf.org/html/rfc3986

ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*
http://isotc.iso.org/livelink/livelink.exe/4230517/ISO_IEC_Directives__Part_2__Rules_for_the_structure_and_drafting_of_International_Standards__2004__5th_edition___pdf_format_.pdf?func=doc.Fetch&nodeid=4230517

SNIA SMI-S, *Storage Management Technical Specification* 1.3
http://www.snia.org/tech_activities/standards/curr_standards/smi/SMI-S_Technical_Position_v1.3.0r5.zip

NOTE: This profile refers to the following clauses of SNIA SMI-S:1.3, Part 2 *Common Profiles*:

Clause 6: *Generic Target Ports* profile 1.0

Clause 14: *Generic Initiator Ports* profile 1.0

This profile refers to the following clauses of SNIA SMI-S:1.3, Part 3 *Block Devices*:

Clause 5: *Block Services* package 1.3

Clause 15: *Extent Composition* subprofile 1.2

This profile refers to the following clauses of SNIA SMI-S:1.3, Part 6 *Host Elements*:

Clause 6: *Storage HBA* profile 1.3

Clause 7: *Host Discovered Resources* profile 1.2

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply. For the purposes of this document, the terms and definitions given in DMTF DSP1033:1.0 (Profile Registration *Profile*) and DMTF DSP1001:1.0 (Management Profile Specification Usage Guide) also apply.

3.1
can
used for statements of possibility and capability, whether material, physical, or causal

3.2
cannot
used for statements of possibility and capability, whether material, physical, or causal

3.3
conditional
indicates requirements strictly to be followed in order to conform to the document and from which no deviation is permitted when the specified conditions are met

3.4
mandatory
indicates requirements strictly to be followed in order to conform to the document and from which no deviation is permitted

3.5
may
indicates a course of action permissible within the limits of the document

3.6
need not
indicates a course of action permissible within the limits of the document

3.7
optional
indicates a course of action permissible within the limits of the document

3.8

referencing profile

indicates a profile that owns the definition of this class and can include a reference to this profile in its "Related Profiles" table

3.9

shall

indicates requirements strictly to be followed in order to conform to the document and from which no deviation is permitted

3.10

shall not

indicates requirements strictly to be followed in order to conform to the document and from which no deviation is permitted

3.11

should

indicates that among several possibilities, one is recommended as particularly suitable, without mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

3.12

should not

indicates that a certain possibility or course of action is deprecated but not prohibited

3.13

unspecified

indicates that this profile does not define any constraints for the referenced CIM element

3.14

client

an application that exploits facilities specified by this profile

3.15

implementation

a set of CIM providers that realize the classes specified by this profile

3.16

this profile

this DMTF management profile – the *Storage Resource Virtualization Profile*

3.17

concrete storage resource pool

a storage resource pool that subdivides the capacity of its (primordial or concrete) parent resource pool

3.18

host storage resource

a storage resource that exists in scope of or is accessible by a host system. A host system may contain or have access to one or more storage resources that may be as a whole or partially allocated to virtual systems

3.19

host system

the scoping system that contains or has access to storage resources that may be allocated, virtualized, or both

3.20

initiator port

a port that acts as the source for a data exchange operation

3.21

logical disk

the instantiation of allocated host resources that is exposed to a virtual system through a storage device; the result of a storage resource allocation based on a storage resource allocation request

3.22

port

communication endpoint for systems or storage devices. A port enables the exchange of data according to one or more protocols

3.23

primordial storage resource pool

a storage resource pool that aggregates storage resources available for or used by storage resource allocations

3.24

storage pool

a special kind of storage resource pool that is managed through SMI-S

3.25

storage resource

a logical disk, a storage volume or a storage extent

3.26

storage resource allocation

the allocation of a storage resource from a storage resource pool to a virtual system

3.27

storage resource allocation request

a request for a storage resource allocation

3.28

storage resource pool

a resource pool that represents storage resources available for storage resource allocation

3.29

storage resource pool configuration service

a configuration service that supports the addition or removal of host storage resources to or from a storage resource pool, and the creation or deletion of concrete subpools of a storage resource pool

3.30

storage volume

the instantiation of allocated host resources that is exposed to a virtual system through a storage device that is published for use outside of the scoping system. Like a logical disk, a storage volume is the result of a storage resource allocation based on a storage resource allocation request

3.31

target port

a port that acts as a target of a data exchange operation

3.32

virtual computer system
virtual system

the concept of virtualization as applied to a computer system

Other common industry terms are *virtual machine*, *hosted computer*, *child partition*, *logical partition*, *domain*, *guest*, or *container*.

3.33

virtualization platform

virtualizing infrastructure provided by a host system that enables the provisioning and deployment of virtual systems

4 Symbols and abbreviated terms

The following symbols and abbreviations are used in this document.

4.1

CIM

Common Information Model

4.2

CIMOM

CIM object manager

4.3

ESD

CIM_ElementSettingData

4.4

HBA

host bus adapter

4.5

RASD

CIM_ResourceAllocationSettingData

4.6

SASD

CIM_StorageAllocationSettingData

4.7

SAN

storage area network

4.8

SDS

CIM_SettingsDefineState

4.9

SLP

Service Location Protocol

4.10
SMI-S
Storage Management Initiative Specification

4.11
SNIA
Storage Networking Industry Association

4.12
VS
virtual system

4.13
VSSD
CIM_VirtualSystemSettingData

4.14
VSSDC
CIM_VirtualSystemSettingDataComponent

5 Synopsis

Profile Name: Storage Resource Virtualization

Version: 1.0.0

Organization: DMTF

CIM Schema Version: 2.21

Central Class: CIM_ResourcePool

Scoping Class: CIM_System

This profile is a component profile that defines the minimum object model needed to provide for the CIM representation and management of the virtualization of storage extents or of disk drives.

Table 1 lists DMTF management profiles on which this profile depends.

Table 1 – Related Profiles

Profile Name	Organization	Version	Relationship	Description
Resource Allocation	DMTF	1.1	Specializes	The abstract profile that describes the virtualization of resources See DMTF DSP1041:1.1.
Allocation Capabilities	DMTF	1.0	Specializes	The abstract profile that describes capabilities for resource allocation and resource mutability See DMTF DSP1043:1.0.
Profile Registration	DMTF	1.0	Mandatory	The profile that specifies registered profiles
Indications	DMTF	1.0	Optional	The profile that specifies indications
Block Services	SNIA	1.3	Optional	The SMI-S package that describes block services

Profile Name	Organization	Version	Relationship	Description
Host Discovered Resources	SNIA	1.2	Optional	The SMI-S profile that describes host discovered resources
Generic Initiator Ports	SNIA	1.0	Optional	The SMI-S profile that describes generic initiator ports
Generic Target Ports	SNIA	1.0	Optional	The SMI-S profile that describes generic target ports

Table 2 lists conditional and optional features defined in this profile.

Table 2 – Optional Features

Feature Name	Requirement Level	Granularity	Description
Resource aggregation	Conditional	Instance of CIM_ResourcePool	The feature that defines the representation of the aggregation of host resources into host resource pools. See 7.5.
Resource pool management	Optional	Instance of CIM_ResourcePool-ConfigurationService	The feature that defines the management of resource pools. See 7.7.

NOTE: Some elements adapted by this profile are defined with a requirement level "conditional", with the condition referring to the implementation of a particular feature. This in effect requires the implementation of the conditional element if the feature is implemented.

6 Description

This clause contains informative text only.

This clause introduces the management domain addressed by this profile, and outlines the central modeling elements established for representation and control of elements in the management domain.

6.1 General

In computer virtualization systems, virtual computer systems are composed of component virtual resources.

This profile specializes the resource virtualization pattern as defined in DMTF DSP1041:1.1 (*Resource Allocation Profile*) and the allocation capabilities pattern as defined in DMTF DSP1043:1.0 (*Allocation Capabilities Profile*) for the representation and management of the following types of resources:

- virtual storage resources, designated by one of the resource type values 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent)
- virtual disk drives, designates by the value 17 (Disk Drive)

This profile references additional CIM elements and establishes constraints beyond those defined in the referenced profiles.

Storage resources represented and managed by means of this profile appear to an operating system running in the virtual computer system as virtual disks. Virtual disks either emulate "physical" disks (such as for example SCSI disks), or appear as "logical" disks that have no physical equivalent (such as for example block devices).

6.2 Storage resource virtualization class schema

Figure 1 shows the class schema of this profile. It outlines the elements that are referenced and in some cases further constrained by this profile, as well as the dependency relationships between elements of this profile and other profiles. For simplicity in diagrams, the prefix *CIM_* has been removed from class and association names. Inheritance relationships are shown only to the extent required in the context of this profile.

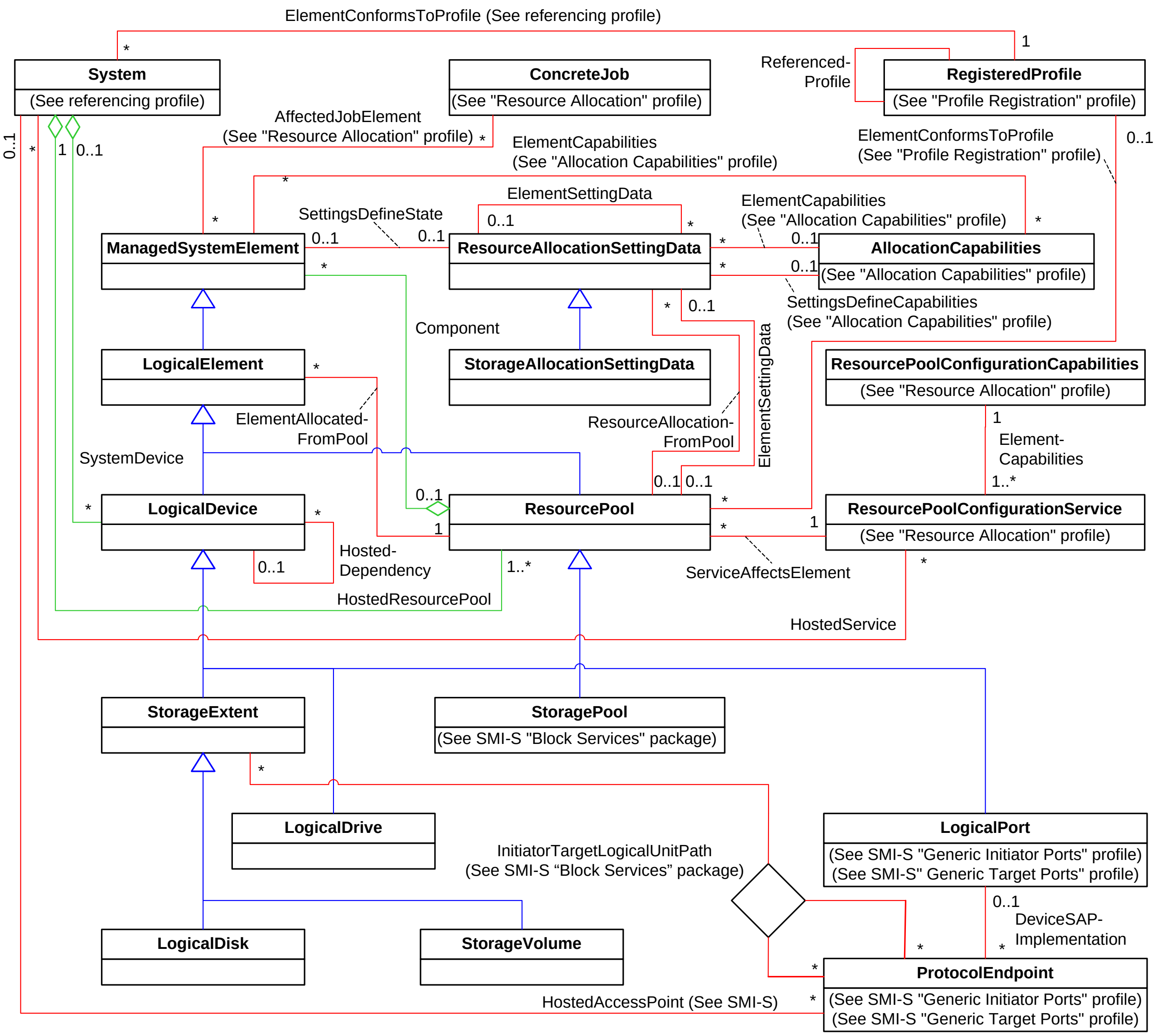

Figure 1 – Storage Resource Virtualization Profile: Profile class diagram

This profile specifies the use of the following classes and associations:

- The CIM_ResourcePool class models resource pools for storage resources (such as storage volumes or logical disks) or for disk drives.
- The CIM_Component association models the relationship between resource pools and host storage resources as components of the resource pools.

- The CIM_ElementAllocatedFromPool association models hierarchies of resource pools and models the relationship of resource pools and storage resources allocated from those.
- The CIM_HostedResourcePool association models the hosting dependency between a resource pool and its host system. A host system supports at least one resource pool for storage resources.
- The CIM_LogicalDisk class, the CIM_StorageVolume class and the CIM_StorageExtent class model the following aspects of logical disks and storage volumes:
  – logical disks, storage volumes or plain storage extents as devices in the scope of a system, as modeled by the CIM_SystemDevice association
  – host storage extents (including subtypes) as components within storage resource pools, as modeled by the CIM_Component association
  – virtual disks as a result of a storage resource allocation from a storage resource pool, as modeled by the CIM_ElementAllocatedFromPool association
- The CIM_DiskDrive class models the following aspects of disk drives:
  – disk drives as devices in the scope of a system, as modeled by the CIM_SystemDevice association
  – disk drives as components within disk drive resource pools, as modeled through the CIM_Component association
  – disk drives as a result of a disk drive allocation from a disk drive resource pool, as modeled by the CIM_ElementAllocatedFromPool association
- The CIM_ResourceAllocationSettingData class models disk drive resource allocations or disk drive resource allocation requests
- The CIM_StorageAllocationSettingData class models storage resource allocations or storage resource allocation requests
- The CIM_AllocationCapabilities class and the CIM_ElementCapabilities association models
  – the resource allocation capabilities of host systems
  – the resource allocation capabilities of storage resource pools
  – the mutability of existing resource allocations
- The CIM_SettingsDefineCapabilities association models the relation between allocation capabilities and the settings that define these capabilities
- The CIM_ResourcePoolConfigurationService class models configuration services for resource pools and the CIM_ResourcePoolConfigurationCapabilities class modeling their capabilities
- The CIM_ConcreteJob class and the CIM_AffectedJobElement association models asynchronous management tasks initiated through resource pool configuration services
- The CIM_HostedDependency association models
  – the relationship between virtual storage extents and host storage extents
  – the relationship between virtual disk drives and host disk drives

6.3 Resource pools

This subclause describes the use of resource pools for storage resources and disk drives.

6.3.1 General

This profile applies the concept of resource pools defined in DMTF DSP1041:1.1, 6.1.2 (Resource Allocation *Profile*) to the following resource types:

- The resource type 31 (Logical Disk) designates storage resource pools that represent resources for the allocation of logical disks for immediate use by virtual systems; allocated logical disks are represented by instances of the CIM_LogicalDisk class
- The resource type 32 (Storage Volume) designates storage resource pools that represent resources for the allocation of storage volumes to virtual storage arrays; allocated storage volumes are represented by the instances of the CIM_StorageVolume class
- The resource type 19 (Storage Extent) designates storage resource pools that represent resources for the allocation of storage extents for virtual systems or virtual storage arrays that are not covered through resource types 31 (Logical Disk) or 32 (Storage Volume) as defined above
- The resource type 17 (Disk Drive) designates virtual disk drive storage pools that represent resources for the allocation of disk drives to virtual systems; allocated disk drives are represented by instances of the CIM_DiskDrive class

Note that the resource type of a resource pool governs the type of the resources that are allocated from the resource pool. Opposed to that the resource type of the resources that are aggregated by the resource pool may differ from the resource type of the pool. For example, a resource pool with a resource type of 31 (Logical Disk) supports the allocation of logical disks. However, the resources that are aggregated by that resource pool may be of a different type; for example, that resource pool might aggregate files, or it might represent a file system without representing individual files.

This profile uses the resource pool as the focal point for storage resource allocations and disk drive allocations. Virtual systems receive storage resource allocations from storage resource pools based on storage resource allocation requests. Virtual systems receive disk drive resource allocations from disk drive resource pools based on disk drive resource allocation requests. In addition, a disk drive may also be allocated as a side effect of a storage resource allocation.

6.3.2 Representation of host resources

A resource pool represents host resources that enable the allocation of virtual devices (such as virtual disks or virtual disk drives). However the explicit representation of the host resources aggregated by a resource pool is optional: In some cases implementations may explicitly represent the host resources such as for example host logical disks, host storage volumes, host disk drives, files, file systems or file directories that are accessible by the host. In other cases implementations may choose not to explicitly represent the host resources aggregated by a resource pool. For example, an implementation that implements the representation and management of memory based virtual disks is not required to explicitly model the host memory that support the virtual disks. Instead, in this case the resource pool is the sole model element that represents host memory capacity assigned for the support of (allocated) virtual disks, and the host capacity that is still available for the allocation of new virtual disks.

DMTF DSP1041:1.1 (*Resource Allocation Profile*) defines two general types of resource pools: Primordial resource pools and concrete resource pools.

NOTE: The SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package provides much stricter definitions of primordial storage pool and concrete storage pool than those of DMTF DSP1041:1.1. Implementations of the SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package for the management of host storage resources need to conform with these definitions. For example, this profile allows the direct use of a primordial resource pool for the allocation of resources, while the SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package supports the creation and modification of storage volumes and logical disks only in context of concrete storage pools.

6.3.3 Primordial resource pool

A primordial resource pool aggregates capacity; it represents a subset of the manageable resources of a host system. Primordial resource pools are suitable to serve as the source of resource allocations — either for the allocation of child resource pools or for virtual resources.

6.3.4 Concrete resource pool

A concrete resource pool subdivides the capacity of its parent resource pool. The amount of capacity allocated to a concrete resource pool is less than or at most equal to the capacity of the parent pool.

6.3.5 Hierarchies of resource pools

This profile specializes the concept of resource pool hierarchies defined in DMTF DSP1041:1.1, 6.1.4 (*Resource Allocation Profile*) to the resource types 31 (Logical Disk), 32 (Storage Volume) and 17 (Disk Drive).

Figure 2 shows an example of the CIM representation of a resource pool hierarchy where a set of host storage extents is aggregated into a primordial storage resource pool PRIM_POOL. The allocation capabilities of PRIM_POOL are represented by means of DMTF DSP1043:1.0 (*Allocation Capabilities Profile*). PRIM_POOL supports allocations starting from 4 GB up to 4 TB; in that range the increment is 1 KB.

Two host storage extents EXTENT1 and EXTENT2 are components of PRIM_POOL. The instances represent storage extents that are available to the host system. For example, the host system itself contains these storage devices such as for example local SCSI disks; or the host system has access to these devices through means of a storage area network or other network mechanisms.

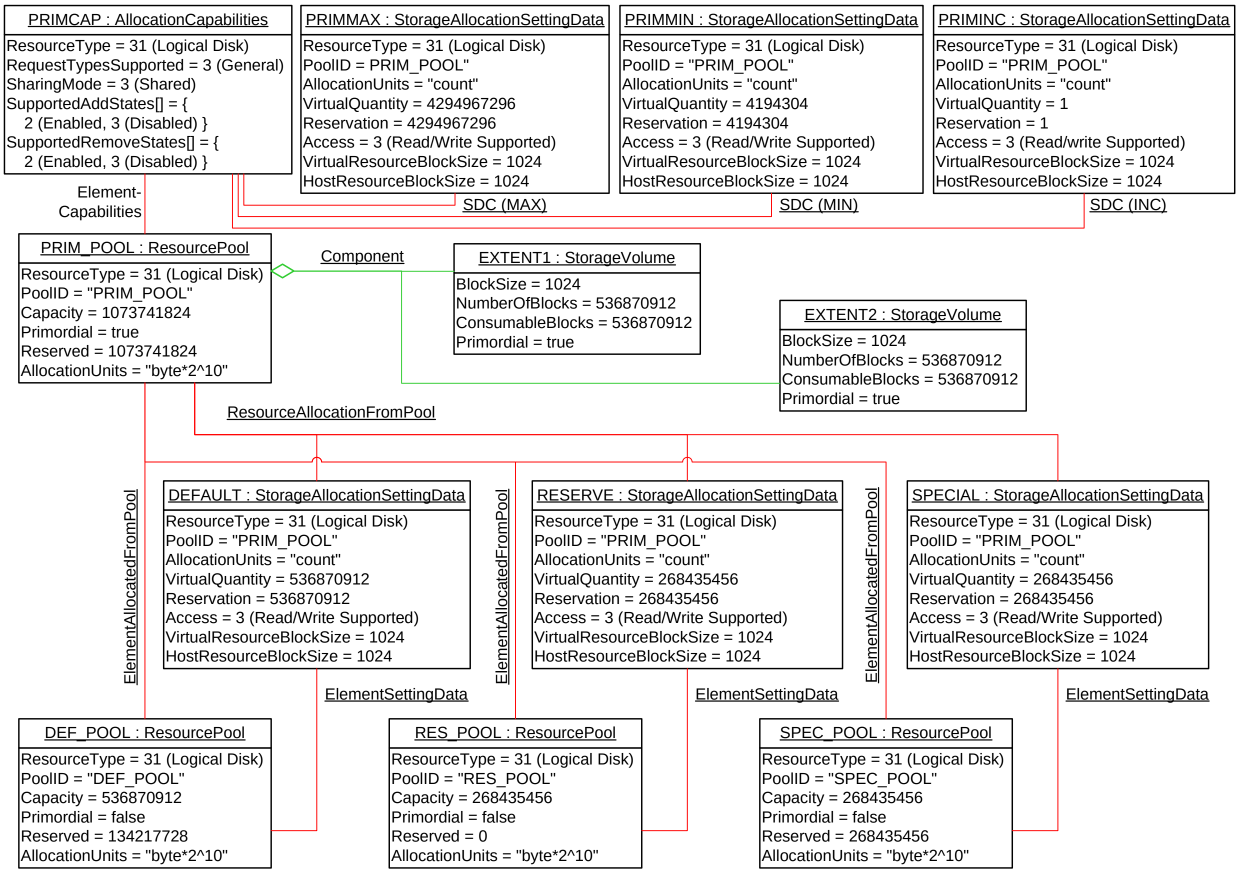

Figure 2 – Instance diagram: Concept of storage resource pool hierarchies

PRIM_POOL is subdivided into three concrete storage resource pools DEF_POOL, RES_POOL and SPEC_POOL, as follows:

- The concrete storage resource pool DEF_POOL represents a subextent of 512 GB. An amount of 128 GB is allocated out of that pool to support virtual disks that are not shown in Figure 2.
- The concrete storage resource pool RES_POOL represents a subextent of 256 GB. No allocations for virtual disks are drawn from this pool in the represented situation.
- The concrete storage resource pool SPEC_POOL represents a subextent of 256 GB. The complete capacity of SPEC_POOL is allocated to support virtual disks that are not shown in Figure 2.

6.3.6 Resource pool management

This profile specializes the concept of resource pool management defined in DMTF DSP1041:1.1, 6.1.5 (*Resource Allocation Profile*) to the resource types 31 (Logical Disk), 32 (Storage Volume), 19 (Storage Extent) and 17 (Disk Drive).

6.4 Resource allocation

This profile specializes the concept of device resource allocation defined in DMTF DSP1041:1.1, 6.3 (*Resource Allocation Profile*) to the resource types 31 (Logical Disk), 32 (Storage Volume), 19 (Storage Extent) and 17 (Disk Drive).

6.4.1 General

Depending on the resource type the result of a resource allocation as seen by a virtual system is either a virtual storage extent (including specializations such as a virtual disk or a virtual storage volume), or a virtual disk drive. In addition, the allocation of a virtual disk or a virtual storage volume may cause the allocation of a virtual disk drive as a side effect.

The representation of disk drives is optional. This profile addresses three potential scenarios:

1) The allocation of a storage extent without explicit representation of a disk drive

2) The allocation of a storage extent with explicit representation of a disk drive

   An example is a virtual disk drive that is based on an image file stored in a host file. The disk drive may be implicitly allocated along with the allocation of the storage extent.

3) The allocation of a disk drive without modeling the allocation of a storage extent

   An example is a disk drive that is owned by a host system and is pathed through to a virtual system; in this case the media is volatile and cannot generally be modeled.

This profile specifies the use of CIM_StorageAllocationSettingData class and the CIM_ResourceAllocationSettingData class such that all of these scenarios are covered.

For example, Figure 3 on page 23 shows a situation like in case 2) above. In this example the allocation of a logical disk to a virtual system causes the implicit allocation of a disk drive. Note that no separate RASD instance is required for allocation of the disk drive. Instead the implementation implicitly allocates the disk drive as part of the allocation of the logical disk and represents the disk drive by an instance of the CIM_DiskDrive class. The CIM_DiskDrive instance is associated to the instance of the CIM_LogicalDisk class representing the allocated logical disk by an instance of the CIM_MediaPresent association.

6.4.2 Storage resource allocations backed by files

In the example shown in Figure 3 the CIM_StorageAllocationSettingData instances directly refer to an image file through the value of the HostResource[ ] array property. In this example the value is formatted as a URI that encodes the file name. The value of the PoolID property refers to a resource pool that in this example represents a source for host files.

Implementations have various choices how to establish the relationship between host files and virtual disks, such as for example:

- Referring to preallocated files in the host environment
- Creation files as a side effect

An example of the latter case is depicted in Figure 3. In this case the implementation established a resource pool such for file-based logical disks. Allocations out of that resource pool are based on host files.

In this example the initial allocation of these host files is controlled by an implementation dependent rule that constructs the file location from several parts:

- a root path (such as for example "/var/vmfiles")
- a virtual system specific subpath (such as for example "vm1/disks" for a virtual system named "vm1"), and
- a disk specific file name (such as for example "imagedisk25.dsk").

The concatenation of the parts yields "/var/vmfiles/vm1/disks/imagedisk25.dsk". Note that such rules are highly implementation dependent. The model defined in this profile facilitates the representation of rule-based assignment through the means of resource pools, but it does not specify elements that explicitly convey information about the rules themselves.

This profile does not require implementations to expose information about resource availability or resource consumption in context of resource pools. For example, in Figure 3 in the CIM_ResourcePool instance FILEDISK_POOL the value of the Capacity property is NULL, indicating that the capacity of the resource pool is unknown to the implementation. Similarly, the value of the Reserved property is NULL, indicating that the amount of consumed resource is unknown to the implementation. It is expected that many implementations will be unaware of the amount of resource available through a resource pool because the resource pool is backed with resources from external units such as storage subsystems and network attached storage. In these situations the only purpose of the resource pool is to represent a particular source for resource allocations without requiring an implementation to have knowledge about resource capacity or consumption. A management client would have to contact the management interface of the external units in order to access respective information.

The example shown in Figure 3 also shows a representation of thin provisioning of a file-based logical disk. This is indicated by the value of the Limit property being defined and higher than the value of the Reservation property. In this example the value of the Reservation property requests an initial file size of 33554432 blocks (16 GB) up to a maximum file size of 134217728 blocks (64 GB) as expressed by the value of the Limit property; in both cases a block size of 512 applies as expressed by the value of the HostResourceBlockSize property. Opposed to that the disk size as seen by the virtual system (the consumer) remains constant at 64 GB; this is expressed by the value of the VirtualQuantity property, 16777216 blocks with a block size of 4096 as expressed by the value of the VirtualResourceBlockSize property. As the consuming virtual system starts writing data onto the virtual disk, for each logical 4-KB block of virtual disk a respective set of eight 512-KB blocks is allocated within the file. As soon as the amount of data to back the logical disk exceeds the initially requested file size (as expressed by the value of the Reservation property) the file starts growing beyond the initially assigned file size up to the size expressed by the value of the Limit property, such that finally when all blocks were at least once written by the virtual system the amount of storage provided equals the amount of storage consumed.

Note that in the example shown in Figure 3 the value of the Limit property expressed an amount of storage that is identical to that expressed by the value of the VirtualQuantity property. This implies that the file may grow until the complete virtual disk is backed with respective file data blocks. However, implementations may support placing restrictions on the file size; for example this may be the case in situations where a specific usage pattern such as a sparsely used disks is expected by the consumer. In this case if the upper file size as expressed by the value of the Limit property is reached, the consumer would receive a respective error indication.

Another concept applied in the example shown in Figure 3 is the remapping of blocks. In this example the block size at the providing host side is 512 (the value of the HostResourceBlockSize property), while the block size at the consuming virtual system side is 4096 (the value of the VirtualResourceBlockSize property).

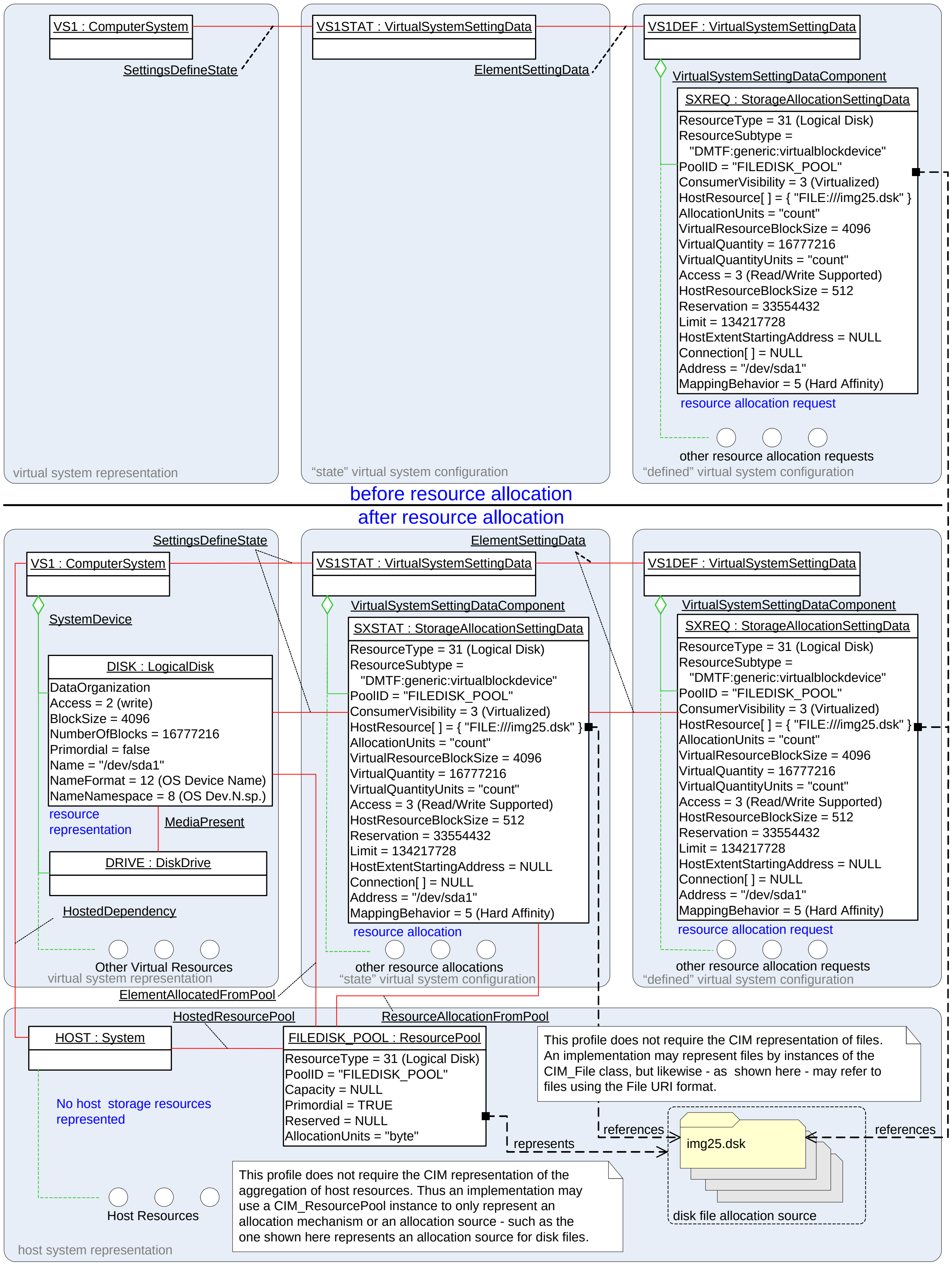

Figure 3 – Instance diagram: Concept of storage resource allocation

6.4.3 Resource allocation request

The resource requirements of a virtual system are represented by the "defined" virtual system configuration (see DMTF DSP1057:1.0 (*Virtual System Profile*)). In a "defined" virtual system configuration disk drive resource allocation requests are represented by CIM_ResourceAllocationSettingData instances, and storage resource allocation requests are represented by CIM_StorageAllocationSettingData instances.

An example of the CIM representation of a storage resource allocation request is shown in the upper right part of Figure 3.

6.4.4 Resource allocation

As a virtual system is activated (instantiated), resources are allocated as requested by resource allocation requests in the "defined" virtual system definition. The actual resource allocations for a virtual system are represented by the "state" virtual system configuration (see DMTF DSP1057:1.0 (*Virtual System Profile*)). In a "state" virtual system configuration disk drive resource allocations are represented by CIM_ResourceAllocationSettingData instances, and storage resource allocations are represented by CIM_StorageAllocationSettingData instances.

NOTE: Storage resource allocation requests and storage resource allocations may directly reference persistent host resources — such as for example host storage extents or host files - through the value of the HostResource[ ] array property. These host resources persistently exist independent of their use as the base for virtual disks. However, there are situations where such host resources are unavailable at resource allocation time. For example, the file system that contains the file referenced by a storage resource allocation request might not be mounted, or a file might be in use by another consumer such as the host system itself or another virtual system. In these situations the resource allocation would fail at resource allocation time.

An example of the CIM representation of a storage resource allocation is shown in the center part of Figure 3.

6.4.5 Virtual disk

A virtual disk is the instantiation of resources allocated from a storage resource pool that is exposed to a virtual system through a logical device; it is the result of a storage resource allocation based on a storage resource allocation request.

Virtual disks may be virtualized or may be passed-through host storage resources.

An example of the CIM representation of a virtualized virtual disk as the result of a storage resource allocation is shown on the left side in the central part of Figure 3.

6.4.6 Virtual disk drive

A virtual disk drive is the instantiation of resources allocated from a resource pool that is exposed to a virtual system through a logical disk drive device; it is either the explicit result of a disk drive resource allocation based on a disk drive resource allocation request, or it is the implicit result of a storage resource allocation based on a storage resource allocation request.

Virtual disk drives may be virtualized or may be passed-through host disk drives. A virtual disk drive is represented by an instance of the CIM_DiskDrive class as part of the virtual system representation.

An example of the CIM representation of a virtual disk drive as the implicit result of a storage resource allocation is shown on the left side in the central part of Figure 3. In this case the virtual disk drive is implicitly allocated as a side effect of a storage resource allocation.

6.4.7 Storage virtualization

In the scope of this profile virtualization of storage is modeled through subdivision: Non overlapping sub-extents of a larger host storage extent may be assigned to different virtual systems. This allows subdividing a host storage extent for the use of a number of virtual systems.

6.4.8 Dedicated host storage

A dedicated storage extent is a storage extent owned or accessible by the host system that is exclusively reserved for support of a particular virtual disk of a particular virtual system.

6.4.9 Virtual storage extended configurability

Some types of virtual storage support extended configuration capabilities. For example virtual SCSI disks emulating physical SCSI disks may be grouped on virtual SCSI buses such that access is channeled from a virtual SCSI initiator port to a virtual SCSI target port, and from there to a target virtual LUN. An implementation may opt to represent ports and LUNs as modeled by respective profiles from SNIA SMIS:1.3 (Storage Management Technical Specification), such as the Generic Initiator Ports profile or the Generic Target Ports profile described in SNIA SMIS:1.3, Part 2 *Common Profiles* book, or more specific profiles that are based on these.

6.4.10 Management of host storage resources through SMI-S profiles

Implementation of this profile may optionally implement profiles defined in SNIA SMIS:1.3 (Storage Management Technical Specification), such as the SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package or the *Extent Composition* subprofile for the management of host storage resources. An example of such a situation is depicted in Figure 4.

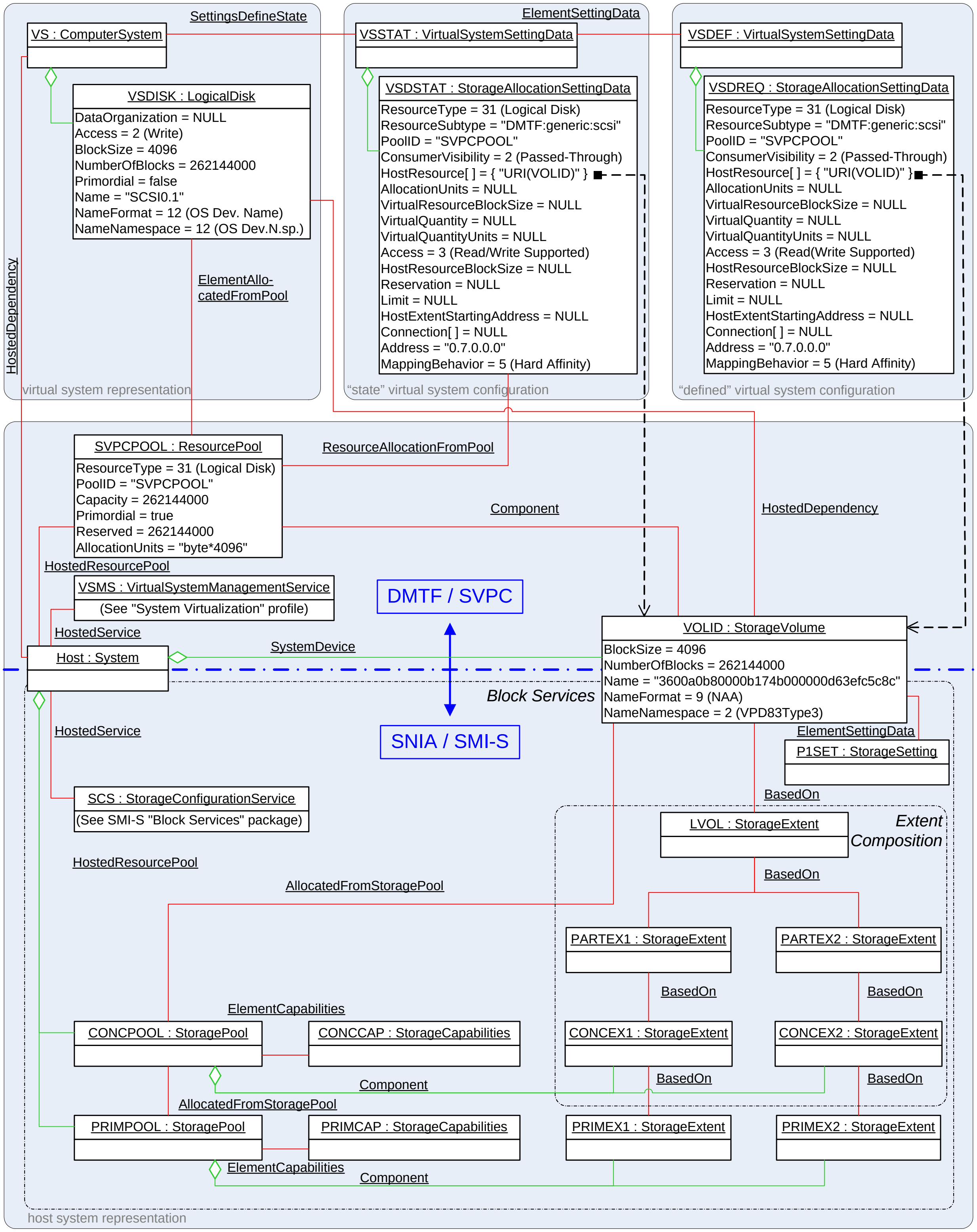

Figure 4 – Cooperation of DMTF SVPC and SNIA SMI-S profiles

This profile (along with other system virtualization related profiles such as DMTF DSP1042:1.0) governs the allocation of storage resources to virtual systems. SMI-S profiles govern the allocation of host storage resources. The boundary between the model defined in this profile and the models defined in SNIA

SMIS:1.3 is defined by CIM_StorageVolume instances such as VOLID in Figure 4. The configuration of these instances into the host environment may be represented and managed by means of SNIA SMIS:1.3, the configuration of virtual disks based on passed-through host disks is represented and managed by means of this profile in coordination with other profiles of the SVPC suite of profiles. However, the use of SMI-S is optional in the context of this profile; respective host resources may just as well be just discovered within the hosting environment by the implementation of this profile.

The upper part of Figure 4 shows the configuration of an instantiated virtual system represented by the CIM_ComputerSystem instance VS1, with a logical disk represented by the CIM_LogicalDisk instance VSDISK. The logical disk is based on a passed-through host disk that is represented by the CIM_StorageVolume instance VOLID. In this case the CIM_StorageAllocationSettingData instances are not required to contain size information because the value of the HostResource[ ] array parameter directly identifies VOLID that is of a given size.

The lower part of Figure 4 shows the configuration of VOLID as it might be presented by an implementation of the SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package. Several layered storage pools in scope of the host system enable the creation and management of block storage resources. The implementation of the *Extent Composition* subprofile on the right side enables the representation of cascaded combinations and / or subdivisions of storage extents.

Note that all aspects managed through SNIA SMIS:1.3 profiles address the representation and management of *host* storage capacity and *host* storage elements. Opposed to that the main functionality specified by this profile is the allocation and management of host resources in support of *virtual* storage resources such as *virtual* disks. In other words, the implementation of profiles from SNIA SMIS:1.3 in combination with an implementation of this profile is supplemental with respect to the representation and management of host storage resources, but not with respect to the allocation and management of virtual storage resources.

The advantage resulting from implementing profiles from SNIA SMIS:1.3 along with implementing this profile is that the implementation of profiles from SNIA SMIS:1.3 enable a more granular management of host resources and storage pools. These host resources and storage pools may subsequently be referenced in instances of the CIM_StorageAllocationSettingData class that describe storage resource allocation requests and storage resource allocations as specified by this profile.

7 Implementation

This clause provides normative requirements related to the arrangement of instances and properties of instances for implementations of this profile.

7.1 Common requirements

The CIM Schema descriptions for any referenced element and its sub-elements apply.

In references to properties of CIM classes that enumerate values the numeric value is normative and the descriptive text following it in parentheses is informative. For example, in the statement "The value of the ConsumerVisibility property shall be 3 (Virtualized)", the value "3" is normative text and "(Virtualized)" is informative text.

Implementations of this profile shall expose an instance of the CIM_RegisteredProfile class as adapted in 10.11 in the Interop namespace. That instance shall be associated with the CIM_RegisteredProfile instance representing the implementation of the scoping profile through an instance of the CIM_ReferencedProfile association as adapted in 10.10. Additional instance requirements specified in DMTF DSP1033:1.0 (*Profile Registration Profile*) may apply.

7.2 Resource types

This subclause specifies the resource types that are addressed by this profile.

7.2.1 General

This profile may be implemented for the allocation of two principal resource types: *Storage extents* or *disk drives*. Note that *logical disks* and *storage volumes* are specializations of storage extents.

7.2.2 Logical disks, storage volumes and storage extents

This subclause provides definitions of the terms logical disk, storage volume and storage extent as well as their CIM representation as applied by this profile. These definitions refine those provided in the CIM schema definitions of the CIM_LogicalDisk class, the CIM_StorageVolume class and the CIM_StorageExtent class and adopt the consistent parts of respective definitions provided in various places of SNIA SMIS:1.3 for the purposes of this profile.

NOTE: The CIM schema definition of the CIM_LogicalDisk class, the CIM_StorageVolume class and the CIM_StorageExtent class as well as various subprofiles of SNIA SMIS:1.3 present definitions of the terms logical disk, storage volume and storage extent. The essence of these definitions is that a storage extent is an abstraction of a range of storage media, that a logical disk is a consumed storage extent and that a storage volume is a storage extent exposed for external use by consumers.

A *storage extent* is a logically contiguous range of logical blocks on some storage media that supports storing and retrieving data. Storage extents shall be represented by CIM_StorageExtent instances, or by instances of subclasses of the CIM_StorageExtent class if the stricter definitions below apply.

A *logical disk* is a specialization of storage extent that is exposed by the virtualization platform to a virtual system for directly consumption. Logical disks shall be represented by CIM_LogicalDisk instances.

A *storage volume* is a specialization of storage extent that is exposed by the host or by a virtual storage array for complete or partitioned use by virtual systems. Storage volumes shall be represented by CIM_StorageVolume instances. This applies likewise to storage volumes exposed by the host or exposed by a virtual storage array.

7.2.3 Disk drives

A disk drive is a media access device; it provides the functionality to access some kind of media. Disk drives shall be represented by CIM_DiskDrive instances.

An implementation of this profile for the allocation of storage extents may allocate disk drives as a side effect of the allocation of storage extents.

7.3 Host resources

This subclause specifies requirements for the representation of host resources.

7.3.1 Host storage volume

The representation of host storage volumes is conditional.

Condition: This profile is implemented for one of the resource types 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent), and the resource aggregation feature (see 7.5) is implemented.

Each host storage volume that is a component of a storage resource pool shall be represented by exactly one CIM_StorageVolume instance as adapted in 10.15. The CIM_StorageVolume instance shall be associated with the CIM_System instance that represents the host system through an instance of the CIM_SystemDevice association, and with the CIM_ResourcePool instance representing the aggregating resource pool through an instance of a subclass of the CIM_Component association as adapted in 10.1.

7.3.2 Host disk drives

The representation of host disk drives is conditional.

Condition: This profile is implemented for the resource type 17 (Disk Drive), and the resource aggregation feature (see 7.5) is implemented.

Each host disk drive volume that is a component of a storage resource pool shall be represented by exactly one CIM_DiskDrive instance as adapted in 10.2. The CIM_DiskDrive instance shall be associated with the CIM_System instance that represents the host system through an instance of the CIM_SystemDevice association, and with the CIM_ResourcePool instance representing the aggregating resource pool through an instance of a subclass of the CIM_Component association as adapted in 10.1.

7.4 Resource pools

This subclause adapts the CIM_ResourcePool class for the representation of storage resource pools and for disk drive resource pool.

7.4.1 General

Implementations of this profile for one of the resource types 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent) may chose to implement the CIM_StoragePool class (which is a subclass of the CIM_ResourcePool class) in place of the CIM_ResourcePool class. The provisions in this subclause apply likewise to implementations of the CIM_ResourcePool class itself, or to implementations of the CIM_StoragePool class if that is implemented instead of the CIM_ResourcePool class. This profile does not adapt properties defined by the CIM_StoragePool class.

NOTE: The SNIA SMIS:1.3, Part 3 Block Devices, *Block Services* package may be implemented for the management of host storage resources; see 6.4.10. Note that the adaptation of the CIM_StoragePool class in the SNIA SMIS:1.3, Part 3 Block Devices, *Block Services* package imposes much stricter implementation requirements for the CIM_StoragePool class than this profile.

7.4.2 ResourceType property

The value of the ResourceType property shall denote the type of resources that are provided by the resource pool, as follows:

- For resource pools supporting only the allocation of logical disks the value of the ResourceType property shall be 31 (Logical Disk).
- For resource pools supporting only the allocation of storage volumes the value of the ResourceType property shall be 32 (Storage Volume).
- For resource pools supporting the allocation of basic storage extents, logical disks or storage volumes the value of the ResourceType property shall be 19 (Storage Extent).

NOTE: See 7.2.2 for a definition of logical disk, storage volume and storage extent.

- For resource pools supporting the allocation of disk drives the value of the ResourceType property shall be 17 (Disk Drive).

7.4.3 ResourceSubType property

The implementation of the ResourceSubType property is optional.

If the ResourceSubType property is implemented, the provisions in this subclause apply.

The value of the ResourceSubType property shall designate a resource subtype. The format of the value shall be as follows: "<org-id>:<org-specific>". The <org-id> part shall identify the organization that defined the resource subtype value; the <org-specific> part shall uniquely identify a resource subtype within the set of subtype defined by the respective organization.

EXPERIMENTAL

Table 3 – Predefined ResourceSubType values

ResourceSubType value	Description
"DMTF:generic:scsi"	Storage device that appears as SCSI device to the guest operating system
"DMTF:generic:ide"	Storage device that appears as IDE device to the guest operating system
"DMTF:generic:virtualblockdevice"	Storage device that appears as generic block device to the guest operating system. NOTE: This definition of block device is system virtualization specific; it does not imply properties of block devices as defined in SNIA SMIS:1.3, Part 3 Block Devices.
"DMTF:ibm:z:3380"	Storage device that appears as IBM 3380 device to the guest operating system
"DMTF:ibm:z:3390"	Storage device that appears as IBM 3390 device to the guest operating system
"DMTF:ibm:z:9336"	Storage device that appears as IBM 9336 device to the guest operating system
"DMTF:ibm:z:FB-512"	Storage device that appears as IBM FB-512 device to the guest operating system
"DMTF:xen:vbd"	Storage device that appears as Xen virtual block device to the guest operating system

An implementation may use the predefined values in Table 3. However implementations are not bound to apply these values; instead, implementation may apply other vendor defined values.

EXPERIMENTAL

The implementation should apply the mechanisms defined in DMTF DSP1043:1.0 (*Allocation Capabilities Profile*) to expose the resource subtypes that are supported by the implementation.

7.4.4 Primordial property

The value of the Primordial property shall be TRUE in any instance of the CIM_ResourcePool class that represents a primordial resource pool. The value of the Primordial property shall be FALSE in any instance of the CIM_ResourcePool class representing a concrete resource pool.

NOTE: See 6.3.3 and 6.3.4 for definitions of primordial and concrete resource pools.

7.4.5 PoolID property

The value of the PoolID property shall enable unique identification of the CIM_ResourcePool instance within the scoping host system.

7.4.6 Reserved property

The implementation of the Reserved property is optional.

If the Reserved property is implemented, its value shall denote the amount of resource that is actually allocated from the resource pool, as follows:

- If the value of the ResourceType property is any of 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent), the value of the Reserved property shall reflect the amount of storage that is allocated from the resource pool, in units as specified by the value of the AllocationUnits property (see 7.4.8).

- If the value of the ResourceType property is 17 (Disk Drive), the value of the Reserved property shall reflect the number of drives that is allocated from the resource pool.

NOTE: For the resource type 17 (Disk Drive), the value of the AllocationUnits property is fixed to "count".

The special value NULL shall be used if the implementation does not have knowledge about the amount of resource allocated from the pool. This may reflect a permanent or a temporary situation.

7.4.7 Capacity property

The implementation of the Capacity property is conditional.

Condition: The resource aggregation feature is implemented; see 7.5.

If the Capacity property is implemented, its value shall reflect the maximum amount of resource that can be allocated from the resource pool, as follows:

- If the value of the ResourceType property is any of 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent), the value of the Capacity property shall reflect the maximum amount of storage that can be allocated from the resource pool, in units as specified by the value of the AllocationUnits property.
- If the value of the ResourceType property is 17 (Disk Drive), the value of the Capacity property shall reflect the maximum number of disk drives that can be allocated from the resource pool.

NOTE: For the resource type 17 (Disk Drive), the value of the AllocationUnits property is fixed to "count".

The special value NULL shall be used if the implementation does not have knowledge about the resource capacity represented by the pool. This may reflect a permanent or a temporary situation.

7.4.8 AllocationUnits property

The value of the AllocationUnits property shall denote the unit of measurement that applies to resource allocations obtained from the resource pool:

- If the value of the ResourceType property is either 31 (Logical Disk), 32 (Storage Volume) or 19 (Storage Extent), the value of the AllocationUnit property shall express the minimum block size that is supported for the type of host storage extent represented by the resource pool. The value shall match "^byte(*([0-9]{1,}|(2|10)\^[0-9]{1,2})){0,1}$".

  NOTE: The regular expression specifies the basic unit "byte". In order to express a minimum block size the basic unit "byte" may be refined with a factor. The factor may be expressed as a plain number (such as "byte*4096"), or may be based on a power of either 2 (such as "byte*2^10" (kibibyte)) or 10 (such as "byte*10^3" (kilobyte)).

- If the value of the ResourceType property is 17 (Disk Drive), the value of the AllocationUnits property shall be "count".

7.4.9 MaxConsumableResource property

The implementation of the MaxConsumableResource property is optional.

If the MaxConsumableResource property is implemented, its value shall reflect the maximum amount of resource that is allocatable to consumers, in units as expressed by the value of the ConsumedResourceUnit property (see 7.4.11).

NOTE: This property describes the consumer side of allocations, as opposed to the providing side that is described by the Capacity property. This allows the representation of resource pools that support over-commitment. For example, a resource pool of the type 31 (Logical Disk) might be able to support virtual disks with an added up virtual quantity of 4 GB, and base that on a file system capacity of 2 GB.

7.4.10 CurrentlyConsumedResource property

The implementation of the CurrentlyConsumedResource property is optional.

If the CurrentlyConsumedResource property is implemented, its value shall reflect the actually allocated amount of resource to consumers, in units as expressed by the value of the ConsumedResourceUnit property (see 7.4.11).

7.4.11 ConsumedResourceUnit property

The implementation of the ConsumedResourceUnit property is conditional.

Condition: The MaxComsumableResource property (see 7.4.9) or the CurrentlyConsumedResource property (see 7.4.10), or both, are implemented.

If the CurrentlyConsumedResource property is implemented, its value shall state the unit that applies to the values of the MaxComsumableResource property and the CurrentlyConsumedResource property; the same rules as for the AllocationUnits property (see 7.4.8) apply.

7.4.12 Instance requirements

Each resource pool shall be represented by a CIM_ResourcePool instance; the provisions of 10.13 apply.

7.5 Resource aggregation feature

The implementation of the resource aggregation feature is conditional.

Condition: The resource pool management feature is implemented; see 7.7.

Granularity: If implemented, the resource aggregation feature may be separately supported for each resource pool.

The preferred feature discovery mechanism is to resolve the CIM_Component association from the CIM_ResourcePool instance to CIM_ManagedElement instances representing aggregated resources of the storage resource pool. If the resulting set of CIM_ManagedElement instances is not empty, the feature is supported.

NOTE: If the result set is empty, the feature may still be supported, but no resources are aggregated at that point in time; however, if ever for a particular resource pool aggregated resources were exposed, then the feature is still supported even if at a later point in time no resources are aggregated.

7.6 Resource pool hierarchies feature

The implementation of the representation of resource pool hierarchies is optional.

Granularity: If implemented, the resource pool hierarchies feature may be separately supported for each resource pool.

If the representation of resource pool hierarchies is implemented, any concrete resource pool shall be represented through an instance of the CIM_ResourcePool class, where all of the following conditions shall be met:

- The value of the Primordial property shall be FALSE.
- The instance shall be associated through an instance of CIM_ElementAllocatedFromPool association to the instance of the CIM_ResourcePool class that represents its parent resource pool.

  NOTE: The SNIA SMIS:1.3, Part 3 *Block Devices*, *Block Services* package requires the implementation of the CIM_ConcreteComponent association for the representation of the same relationship if the SNIA SMIS:1.3, Part 3 *Block Devices*, *Extent Composition* subprofile is implemented; in this case implementations of the SNIA SMIS:*1.3*, Part 3 *Block Devices*, *Block Services* package need to implement both associations.

- The instance shall be associated through an instance of the CIM_ElementSettingData association to the instance of the CIM_ResourceAllocationSettingData class that represents the amount of resource allocated from the parent pool.

The preferred feature discovery mechanism is to resolve the CIM_ElementAllocatedFromPool association from a CIM_ResourcePool instance to other (parent or child) CIM_ResourcePool instances. If the resulting set of CIM_ResourcePool instances is not empty, the feature is supported; otherwise, the feature is not supported.

NOTE: If for example the Associators( ) intrinsic operation is used to resolve the association, the Role parameter or the ResultRole parameters may be used to distinguish the parent-to-child relationship from the child-to-parent relationship.

7.7 Resource pool management feature

The implementation of the resource pool management feature is optional.

If implemented, the specifications of DMTF DSP1041:1.1 (*Resource Allocation Profile*) apply; this profile does not specify specializations or extensions of resource pool management beyond those defined by DMTF DSP1041:1.1.

7.8 Resource allocation

This subclause details requirements for the representation of resource allocation information through CIM_ResourceAllocationSettingData (RASD) instances or CIM_StorageAllocationSettingData (SASD) instances.

7.8.1 General

Implementations of this profile shall implement the virtual resource allocation pattern as defined in DMTF DSP1041:1.1, subclause 7.2 (Virtual resource allocation).

NOTE: DMTF DSP1041:1.1 specifies two alternatives for modeling resource allocation: *simple resource allocation* and *virtual resource allocation*.

This profile adapts the CIM_StorageAllocationSettingData (SASD) class for storage resource allocation and the CIM_ResourceAllocationSettingData class for disk drive resource allocation.

7.8.2 Flavors of allocation data

Various flavors of allocation data describes are defined:

- Resource allocation requests; for details see 6.4.3.
- Resource allocations; for details see 6.4.4.
- Settings that define the capabilities or mutability of managed resources. DMTF DSP1043:1.0 specifies a capabilities model that conveys information about the capabilities and the mutability of managed resources in terms of RASD instances.
- Parameters in operations that define or modify any of the representations listed above. DMTF DSP1042:1.0 that specifies methods for the definition and modification of virtual resources. These methods use RASD instances for the parameterization of resource-allocation-specific properties.

Table 4 lists acronyms that are used in subclauses of 7.8 in order to designate RASD or SASD instances that represent various flavors of allocation data.

Table 4 – Acronyms for RASD adapted for the representation of various flavors of allocation data

Acronym	Flavor
Q_RASD	RASD adapted for the representation of disk drive resource allocation requests
Q_SASD	SASD adapted for the representation of storage resource allocation requests
R_RASD	RASD adapted for the representation of disk drive resource allocations
R_SASD	SASD adapted for the representation of storage resource allocations
C_RASD	RASD adapted for the representation of settings that define capabilities of systems or disk drive resource pools, or that define the mutability of disk drive allocations or disk drive allocation requests
C_SASD	SASD adapted for the representation of settings that define capabilities of systems or storage resource pools, or that define the mutability of storage resource allocations or storage resource allocation requests
D_RASD	RASD adapted for the representation of new disk drive resource allocation requests in method parameter values
D_SASD	SASD adapted for the representation of new storage resource allocation requests in method parameter values
M_RASD	RASD adapted for the representation of modified disk drive resource allocations or disk drive resource allocation request in method parameter values
M_SASD	SASD adapted for the representation of modified storage resource allocations or storage resource allocation request in method parameter values

Subclauses of 7.8 detail implementation requirements for property values in RASD instances. In some cases requirements only apply to a subset of the flavors listed in Table 4; this is marked in the text through the use of respective acronyms.

7.8.3 CIM_ResourceAllocationSettingData properties

This subclause defines rules for values of properties in instances of the CIM_ResourceAllocationSettingData (RASD) class representing disk drive allocation information.

7.8.3.1 ResourceType property

The value of the ResourceType property in RASD instances representing disk drive allocation information shall be set to 17 (Disk Drive) for disk drive allocation data.

Other values shall not be used.

7.8.3.2 ResourceSubType property

The implementation of the ResourceSubType property is optional.

If the ResourceSubType property is implemented, the provisions defined for the ResourceSubType property of the CIM_ResourcePool class; see 7.4.2.

7.8.3.3 PoolID property

The value of the PoolID property shall identify the resource pool. The special value NULL shall indicate the use of the host system's default resource pool for the selected resource type.

7.8.3.4 ConsumerVisibility property

The implementation of the ConsumerVisibility property is optional.

If the ConsumerVisibility property is implemented, the provisions in this subclause apply.

The value of the ConsumerVisibility property shall denote either if a host resources is directly passed through to the virtual system as a virtual resource, or if the resource is virtualized. Values shall be set as follows:

- A value of 2 (Passed-Through) shall denote that the host resource is passed-through.
- A value of 3 (Virtualized) shall denote that the virtual resource is virtualized.
- Only in instances of { Q_RASD | D_RASD | M_RASD }, the special value NULL shall be used if the represented resource allocation request does not predefine which kind of consumer visibility (passed-through or virtualized) is requested.
- Other values shall not be used.

7.8.3.5 HostResource[ ] array property

The implementation of the HostResource[ ] array property is conditional.

Condition: The HostResource[ ] array property shall be implemented if any of the following conditions is true: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property, or any of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the MappingBehavior property.

If the HostResource[ ] array property is implemented, the provisions in this subclause apply.

In the cases of { Q_RASD | C_RASD | D_RASD | M_RASD } the value of the HostResource[ ] array property shall refer to the representation of one or more host resources that are configured to contribute to the disk drive resource allocation. In the case of R_RASD the value of the HostResource[ ] array property shall refer to a representation of the host resource that provides the disk drive resource allocation.

Elements of the value of the HostResource[ ] array property shall refer to instances of CIM classes, using the WBEM URI format as specified by DMTF DSP0207. Referenced instances shall be of the CIM_CDROMDrive class, the CIM_DiskDrive class, the CIM_DisketteDrive class, the CIM_DVDDrive class or the CIMWORMDrive class.

7.8.3.6 AllocationUnits property

The value of the AllocationUnits property shall be "count".

NOTE: The units defined by value of the AllocationUnits property applies to the values of the Reserved and the Limit property; it does not apply to the value of the VirtualQuantity property.

7.8.3.7 VirtualQuantity property

The value of the VirtualQuantity property shall denote the number of virtual disk drives available to a virtual system through this resource allocation.

EXPERIMENTAL

7.8.3.8 VirtualQuantityUnits property

The value of the VirtualQuantityUnits property shall be "count".

EXPERIMENTAL

7.8.3.9 Reservation property

The implementation of the Reservation property is optional.

If the Reservation property is implemented, the provisions in this subclause apply.

The value of the Reservation property shall denote the number of disk drives reserved through this resource allocation to a virtual system.

7.8.3.10 Limit property

The implementation of the Limit property is optional.

If the Limit property is implemented, the following rules apply:

The value of the Limit property shall denote the maximum number of disk drives available through this resource allocation to a virtual system.

7.8.3.11 Weight property

The implementation of the Weight property is optional.

If the Weight property is implemented, its value shall denote the relative priority of a resource allocation in relation to other resource allocations.

7.8.3.12 Parent property

The implementation of the Parent property is optional.

If the Parent property is implemented, the provisions in this subclause apply.

The value of the Parent property shall identify the parent entity of the resource allocation or resource allocation request. The value of the Parent property shall be formatted with the WBEM URI format as specified by DMTF DSP0207.

If an implementation implements the concept of disk snapshots where data stored on a delta disk only contains information on top of that stored on a base disk, then the implementation should use the value of the Parent property in the RASD instance representing the storage resource allocation of the delta disk to refer to the RASD instance representing the storage resource allocation of the base disk.

7.8.3.13 Connection[ ] array property

The implementation of the Connection[ ] array property is optional.

If the Connection[ ] array property is implemented, the provisions in this subclause apply.

The value of the connection property may identify elements of the storage infrastructure such as initiator ports and/or target ports. The WBEM URI format (see DMTF DSP0207) may be used to refer to a respective CIM instance.

7.8.3.14 Address property

The implementation of the Address property is optional.

If the Address property is implemented, the provisions in this subclause apply.

The value of the Address property shall expose the address of the allocated resource as seen by the software running in the virtual system (usually a guest operating system).

7.8.3.15 MappingBehavior property

The implementation of the MappingBehavior property is optional.

If the MappingBehavior property is implemented, its value shall denote how host resources referenced by elements in the value of HostResource[ ] array property relate to the resource allocation.

In R_RASD instances the following rules apply to the value of the MappingBehavior property:

- A value of 2 (Dedicated) shall indicate that the represented resource allocation is provided by host resources that are exclusively dedicated to the virtual system. The host resources shall be identified by the value of the HostResource[ ] array property.
- A value of 3 (Soft Affinity) or 4 (Hard Affinity) shall indicate that the represented resource allocation is provided by host resources. The host resources shall be identified by the value of the HostResource[ ] array property.
- Other values shall not be used.

In Q_RASD instances the following rules apply to the value of the MappingBehavior property:

- The special value NULL shall indicate that the resource allocation request does not require specific host resources.
- A value of 2 (Dedicated) shall indicate that the resource allocation request shall be provided by exclusively dedicated host resources as specified through the value of the HostResource[ ] array property.
- A value of 3 (Soft Affinity) shall indicate that the resource allocation request shall preferably be provided by host resources as specified through the value of the HostResource[ ] array property, but that other host resources may be used if the requested host resources are not available.
- A value of 4 (Hard Affinity) shall indicate that the resource allocation request shall be provided by host resources as specified through the value of the HostResource[ ] array property and that other resources shall not be used if the requested host resources are not available.
- Other values shall not be used.

7.8.4 CIM_StorageAllocationSettingData properties

This subclause defines rules for values of properties in instances of the CIM_StorageAllocationSettingData (SASD) class.

NOTE: If the rules for a particular property are the same as those defined for the respective property of the CIM_ResourceAllocationSettingData (RASD) class, the respective subclause of 7.8.3 is referenced.

7.8.4.1 ResourceType property

The value of the ResourceType property shall be set as follows:

- 31 (Logical Disk) for logical disk allocation data
- 32 (Storage Volume) for storage volume allocation data

- 19 (Storage Extent) for storage extent allocation data

NOTE: See 7.2.2 for a definition of logical disk, storage volume and storage extent.

7.8.4.2 ResourceSubtype property

See 7.8.3.2.

7.8.4.3 PoolID property

See 7.8.3.3.

7.8.4.4 ConsumerVisibility property

See 7.8.3.4.

7.8.4.5 HostResource array property

The implementation of the HostResource[ ] array property is conditional.

Condition: The HostResource[ ] array property shall be implemented if any of the following conditions is true: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property, or any of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the MappingBehavior property.

If the HostResource[ ] array property is implemented, the provisions in this subclause apply.

In the cases of { Q_SASD | C_SASD | D_SASD | M_SASD } the value of the HostResource[ ] array property shall refer to (the representation of) one or more host resources that are configured to contribute to the resource allocation. In the case of R_SASD the value of the HostResource[ ] array property shall refer to (the representation of) the host resource that provides the storage resource allocation.

Values of elements of the HostResource[ ] array property may directly refer to files, using the URI format as specified by IETF RFC1738 and file URL scheme as specified in IETF RFC3986.

If the file URI is not applied, elements of the value of the HostResource[ ] array property shall refer to instances of CIM classes, using the WBEM URI format as specified by DMTF DSP0207. Referenced instances shall be of the CIM_StorageExtent class or the CIM_LogicalFile class.

7.8.4.6 AllocationUnits property

The implementation of the AllocationUnits property is conditional.

Condition: The Reservation property (see 7.8.4.9) or the Limit property (see 7.8.4.10), or both are implemented.

The AllocationUnits property shall convey the unit applicable to the values of the Reservation and the Limit property.

If the AllocationUnits property is implemented, the provisions in this subclause apply.

If the value of the BlockSize property is 1, the value of the AllocationUnits property shall be "byte", indicating the that the values of the Reservation and of the Limit property are specified in bytes. If the value of the BlockSize property is greater than 1, the value of the AllocationUnits property shall be "count", indicating that the values of the Reservation and of the Limit property are specified in blocks, with the blocksize conveyed through the value of the BlockSize property.

All flavors of SASD instances as defined in 7.8.2 that relate to the same virtual resource shall apply the same value for the AllocationUnits property.

NOTE: The definitions in this subclause include SASD instances that describe mutability. In these instances the mutability is expressed by values of numerical properties such as the Reservation or the Limit property in units as established by the value of the AllocationUnit property. If the mutability SASD instance represents an increment, this would reflect a granularity for modifications of the numeric property values that is equal to or a multiple of the allocation unit.

7.8.4.7 VirtualQuantity property

In the cases of { R_SASD | C_SASD | D_SASD | M_SASD } the value of the VirtualQuantity property shall denote the amount of storage that is available to a virtual system through this resource allocation. In the case of Q_SASD the value of the VirtualQuantity property shall denote the amount of storage that is requested for the virtual system unless the value of the HostResource[ ] array property contains exactly one element that refers to a specific host storage resource that implicitly determines the virtual disk size. If a value is provided, is shall be expressed in units as expressed by the value of the VirtualQuantityUnit property; see 7.8.4.8.

EXPERIMENTAL

7.8.4.8 VirtualQuantityUnits property

The VirtualQuantityUnits property shall convey the unit applicable to the value of the VirtualQuantity property.

If the value of the VirtualResourceBlockSize property is 1, the value of the VirtualQuantityUnits property shall be "byte", indicating the that the value of the VirtualQuantity property is specified in bytes. If the value of the VirtualBlockSize property is greater than 1, the value of the VirtualQuantityUnits property shall be "count", indicating that the value of the VirtualQuantity property is specified in blocks, with the blocksize conveyed through the value of the VirtualResourceBlockSize property.

EXPERIMENTAL

7.8.4.9 Reservation property

The implemenation of the Reservation property is optional.

If the Reservation property is implemented, the provisions in this subclause apply.

The value of the Reservation property shall denote the amount of storage reserved through this resource allocation to a virtual system in units as expressed by the value of the AllocationUnits property; see 7.8.4.6.

7.8.4.10 Limit property

The implementation of the Limit property is optional.

If the Limit property is implemented, the provisions in this subclause apply.

The value of the Limit property shall denote the maximum amount of storage available through the represented resource allocation to a virtual system in units as expressed by the value of the AllocationUnits property; see 7.8.4.6.

7.8.4.11 Weight property

See 7.8.3.11.

7.8.4.12 Parent property

See 7.8.3.12.

7.8.4.13 Connection[ ] array property

See 7.8.3.13.

7.8.4.14 Address property

See 7.8.3.14.

7.8.4.15 MappingBehavior property

See 7.8.3.15.

7.8.4.16 VirtualResourceBlockSize

The value of the VirtualResourceBlockSize property shall denote the block size as seen by the consumer of a virtual storage that is based on the described resource allocation. A value of 1 shall designate a variable block size.

7.8.4.17 Access

The value of the Access property shall denote the access mode.

7.8.4.18 HostResourceBlockSize

The value of the HostResourceBlockSize property shall denote the block size as seen by the consumer of a virtual storage that is based on the described resource allocation. A value of 1 shall designate a variable block size.

7.8.4.19 HostExtentStartingAddress

The implementation of the HostExtentStartingAddress property is optional.

If the HostExtentStartingAddress property is implemented, the provisions in this subclause apply.

The value of the HostExtentStartingAddress property shall denote the offset within the host storage extent referenced by the value of the HostExtentName property. The offset marks the starting point of a subspace within the referenced host storage extent. The size of the subspace is exposed by the value of the Reserved property.

7.8.4.20 HostExtentName

The implementation of the HostExtentName property is optional.

If the HostExtentName property is implemented, the provisions in this subclause apply.

The value of the HostExtentName shall identify a host storage extent that serves as the base for the described virtual storage allocation.

7.8.4.21 HostExtentNameFormat

The implementation of the HostExtentNameFormat property is conditional.

Condition: The HostExtentName property (see 7.8.4.20) is implemented.

If the HostExtentNameFormat property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameFormat shall designate the format used for the value of the HostExtentName property.

7.8.4.22 OtherHostExtentNameFormat

The implementation of the HostExtentNameFormat property is conditional.

Condition: The HostExtentNameFormat property (see 7.8.4.21) is implemented, and the value 1 (Other) is supported.

If the OtherHostExtentNameFormat property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameFormat shall designate the format used for the value of the HostExtentName property, using a string representation. The value should be structured as follows: <Organization>:<FormatSpecifier>. <Organization> shall uniquely identify the organization that defined the format. <FormatSpecifier> shall uniquely identify the format within the set of formats defined by the organization.

7.8.4.23 HostExtentNameNamespace

The implementation of the HostExtentNameNamespace property is conditional.

Condition: The HostExtentName property (see 7.8.4.20) is implemented.

If the HostExtentNameNamespace property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameNamespace shall designate the namespace that applies to the value of the HostExtentName property.

7.8.4.24 OtherHostExtentNameNamespace

The implementation of the OtherHostExtentNameNamespace property is conditional.

Condition: The HostExtentNameNamespace property (see 7.8.4.23) is implemented, and the value 1 (Other) is supported.

If the OtherHostExtentNameNamespace property is implemented, the provisions in this subclause apply.

The value of the HostExtentNameNamespace shall designate the namespace used for the value of the HostExtentName property, using a string representation. The value should be structured as follows: <Organization>:<NamespaceSpecifier>. <Organization> shall uniquely identify the organization that defined the format. <NamespaceSpecifier> shall uniquely identify the namespace within the set of namespaces defined by the organization.

7.8.5 Instance requirements

This subclause details resource allocation related instance requirements.

7.8.5.1 Representation of resource allocation requests

If this profile is implemented for the allocation of storage extents (see 7.2), each storage resource allocation request shall be represented by a Q_SASD instance; the provisions of 10.15 apply.

If this profile is implemented for the allocation of disk drives (see 7.2), each disk drive resource allocation request shall be represented by a Q_RASD instance; the provisions of 10.12 apply.

7.8.5.2 Representation of resource allocations

If this profile is implemented for the allocation of storage extents (see 7.2), each storage resource allocation shall be represented by a R_SASD instance; the provisions of 10.15 apply.

If this profile is implemented for the allocation of disk drives (see 7.2), each disk drive resource allocation shall be represented by a R_RASD instance; the provisions of 10.12 apply.

The R_SASD (or R_RASD) instance shall be associated to the Q_SASD (or Q_RASD) instance representing the corresponding resource allocation request (see 7.8.5.1) through an instance of the CIM_ElementSettingData association; the provisions of 10.6 apply.

The R_SASD (or R_RASD) instance shall be associated to the CIM_ResourcePool instance providing resources for the allocation (see 7.4) through an instance of the CIM_ResourceAllocationFromPool association; the provisions of 10.6 apply.

Implementations may represent a resource allocation request and the corresponding resource allocation by one SASD (or RASD) instance; in this case the association requirements of this subclause apply correspondingly. Note that association instances that refer to the R_SASD instance are only existent while the resource is allocated.

7.8.5.3 Representation of resource allocation capabilities

The allocation capabilities of a system or a resource pool shall be represented by an CIM_AllocationCapabilities instance that is associated to the CIM_System instance representing the system or to the CIM_ResourcePool instance representing the resource pool through an instance of the CIM_ElementCapabilities association; see DMTF DSP1043:1.0 (*Allocation Capabilities Profile*).

The settings that define the allocation capabilities of a storage resource pool shall be represented by C_SASD instances; the provisions of 10.15 apply.

The settings that define the allocation capabilities of a disk drive resource pool shall be represented by C_RASD instances; the provisions of 10.12 apply.

7.8.5.4 Representation of resource allocation mutability

The mutability of a resource allocation or resource allocation request shall be represented by an CIM_AllocationCapabilities instance that is associated to the RASD instance representing the resource allocation of resource allocation request through an instance of the CIM_ElementCapabilities association; see DMTF DSP1043:1.0 (*Allocation Capabilities Profile*).

The settings that define the allocation capabilities of a storage resource pool shall be represented by C_SASD instances; the provisions of 10.15 apply.

The settings that define the allocation capabilities of a disk drive resource pool shall be represented by C_RASD instances; the provisions of 10.12 apply.

7.9 Virtual resources

This subclause specifies rules for the representation of virtual resources. Virtual resources are the result of a resource allocations. Virtual resources are scoped by virtual systems or by virtual storage arrays. Virtual storage arrays are a special kind of virtual system that serve the purpose of providing storage to other virtual systems.

7.9.1 Virtual resource instance requirements

An allocated virtual resource shall be represented by an instance of a subclass of the CIM_LogicalDevice class, as follows:

- Virtual disks

  The representation of virtual disks is governed by the type of virtual disk as defined in 7.2.2.

  – An allocated virtual disk shall be represented by a CIM_StorageExtent instance if the value of the ResourceType property in the CIM_StorageAllocationSettingData instance representing the virtual disk allocation is 19 (Storage Extent). However, if the allocated virtual disk conforms to the stricter definitions of logical disk or storage volume (see 7.2.2), it may be represented by a CIM_LogicalDisk or a CIM_StorageVolume, respectively.

  – A allocated virtual disk shall be represented by a CIM_StorageVolume instance if the value of the ResourceType property in the CIM_StorageAllocationSettingData instance representing the virtual disk allocation is 32 (Storage Volume)
  – A allocated virtual disk shall be represented by a CIM_LogicalDisk instance if the value of the ResourceType property in the CIM_StorageAllocationSettingData instance representing the virtual disk allocation is 31 (Logical Disk).

- Virtual disk drives

  A virtual disk drive shall be represented by an instance of the CIM_DiskDrive class

- Virtual ports

  If implemented, virtual initiator ports or virtual target ports shall be represented by instances of the CIM_LogicalPort class

Each instance of a subclass of the CIM_LogicalDevice class representing a virtual resource as defined in this subclause shall be associated as follows:

- to the CIM_ComputerSystem instance that represents the virtual system through an instance of the CIM_SystemDevice association
- to the SASD or RASD instance that represents the resource allocation through an instance of the CIM_SettingsDefineState association

  NOTE: If the resource allocation of a logical disk is a composed resource allocation, only the top-most resource allocation yields a logical device. Consequently there may be SASD instances within a "state" virtual system configuration that do not have a companion logical device.

- to the CIM_ResourcePool instance that represents the resource pool providing the resource allocation through an instance of the CIM_ElementAllocatedFromPool association.

7.9.2 CIM_StorageExtent properties

This subclause defines constraints for property values in CIM_StorageExtent instances representing virtual disks.

7.9.2.1 BlockSize

The value of the BlockSize property shall be identical to the value of the VirtualResourceBlockSize property in the SASD instance representing the related storage resource allocation (see 7.8.4.16).

7.9.2.2 NumberOfBlocks

The value of the NumberOfBlocks property shall be identical to the value of the VirtualQuantity property in the SASD instance representing the related storage resource allocation (see 7.8.4.7).

7.9.2.3 Name

The value of the Name property shall expose the name of the storage extent as seen by the virtual system.

7.9.2.4 NameFormat

The value of the NameFormat property shall be 12 (OS Device Name).

7.9.2.5 NameNamespace

The value of the NameFormt property shall be 8 (OS Device Namespace).

8 Methods

This section details the requirements for supporting intrinsic operations and extrinsic methods for the CIM elements defined by this profile.

8.1 Profile conventions for operations

The implementation requirements on intrinsic operations for each profile class (including associations) are specified in class specific subclauses of this clause.

The default list of intrinsic operations for all classes is:

- GetInstance( )
- EnumerateInstances( )
- EnumerateInstanceNames( )

For classes that are referenced by an association, the default list also includes

- Associators( )
- AssociatorNames( )
- References( )
- ReferenceNames( )

Implementation requirements on operations defined in the default list are provided in the class-specific subclauses of this clause.

The implementation requirements for intrinsic and extrinsic methods of classes listed in clause 10, but not addressed by a separate subclause of this clause are specified by the "Methods" clauses of respective base profiles, namely DMTF DSP1041:1.1 (*Resource Allocation Profile*) and DMTF DSP1043:1.0 (*Allocation Capabilities Profile*). These profiles are specialized by this profile, and in these cases this profile does not add method specifications beyond those defined in its base profiles.

8.2 CIM_DiskDrive for host disk drives

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.3 CIM_DiskDrive for virtual disk drives

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.4 CIM_LogicalDisk for virtual disk drives

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.5 CIM_ReferencedProfile

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.6 CIM_RegisteredProfile

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.7 CIM_StorageAllocationSettingData for storage extent allocation information

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200:1.3. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.8 CIM_StorageExtent for virtual disk

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.9 CIM_SystemDevice for host storage volumes

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

8.10 CIM_SystemDevice for virtual resources

All intrinsic operations in the default list in 8.1 shall be implemented as specified by DMTF DSP0200. In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles) apply.

9 Use cases

This clause contains informative text only.

The following use cases and object diagrams illustrate use of this profile. They are for informative purposes only and do not introduce behavioral requirements for implementations of the profile.

9.1 Instance diagram

Figure 5 depicts the CIM representation of a host system with one storage resource pool and one virtual system. Only information relevant in the context of storage resource virtualization is shown.

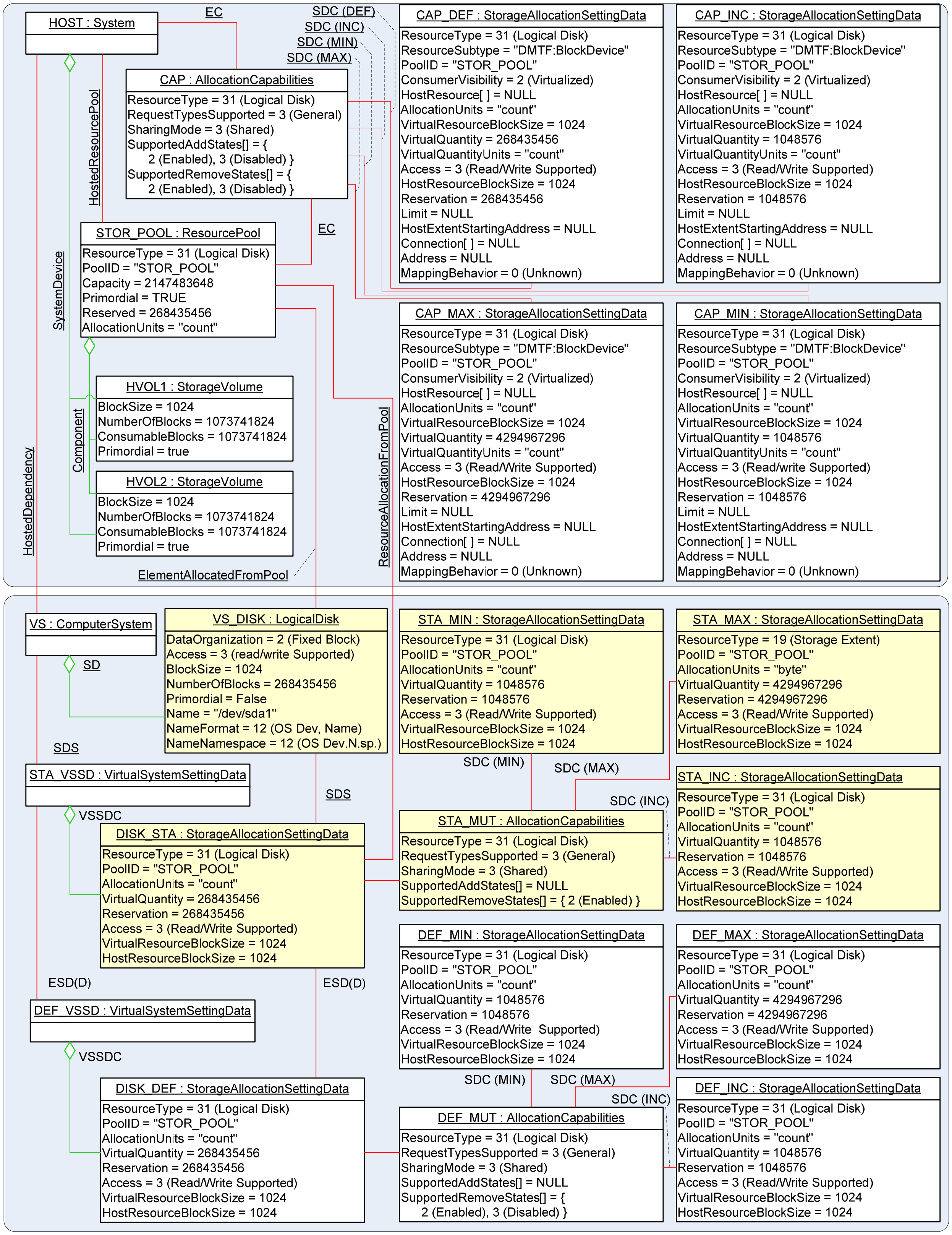

Figure 5 – Instance diagram: Example CIM representation of storage resource virtualization

In Figure 5 the host system is represented by an instance HOST of the CIM_System class. The host system owns or has access to two storage volumes each with a size of 1TB that are represented by the CIM_StorageExtent instances HVOL1 and HVOL2. Note that the storage volumes may be located within a storage area network that is not part of the host system itself.

The host system hosts a primordial storage resource pool that is represented by the CIM_ResourcePool instance STOR_POOL. The value of the ResourceType property in STOR_POOL is 31 (Logical Disk), designating the type of resources that are allocated out of the resource pool.

The resource type of resources aggregated by a resource pool may be different from the type of resources allocated from the pool. In this example as shown in Figure 5 both host storage volumes are aggregated into the pool, as represented by CIM_Component instances connecting STOR_POOL with HVOL1 and HVOL2, respectively.

In the example shown in Figure 5 the storage allocation capabilities of the host system and of the storage resource pool are identical and represented by the same CIM_AllocationCapabilities instance CAP. Four SASD instances (CAP_DEF, CAP_MIN, CAP_MAX, and CAP_INC) are associated with CAP through CIM_SettingsDefineCapabilities (SDC) instances. Not shown Figure 5 are the values of the ValueRange and ValueRole properties in the SDS instances that designate the referenced SASD instances as representing the default, minimum, maximum, and increment for storage resource allocations that are supported by the system and the pool; instead the lines depicting the association instances are respectively labeled as SDC(DEF, SDC(MIN), SDC(INC) and (SDC(MAX). The values of the VirtualQuantity property in the CAP_xxx instances indicate that virtual disks allocatable from the resource pool have a minimum supported size of 1 MB up to a maximum supported size of 4TB, and that an increment of 1 MB applies within the supported range; the default size is 256 GB.

In the CAP_xxx instances the value of the AllocationUnit property is "count"; this indicates that the value of the Reservation property is expressed in blocks. The block size is exposed by the value of the HostResourceBlockSize property (1024). Consequently storage allocations as seen from the providing host system or resource pool side are expressed in 1-KB blocks.

Similarly, the value of the VirtualQuantityUnits property is "count", indicating that the value of the VirtualQuantity property is expressed in blocks. Here the block size is exposed by the value of the VirtualResourceBlockSize property (1024). Consequently storage allocations as seen by the consuming virtual system are expressed in 1-KB blocks as well.

Note that the CAP_xxx instances do not expose a value for the Limit property. This indicates that the implementation does not support thin provisioning where the resource on the consuming side appears larger than the amount of resource provided at the providing side. This implies that the values of the numeric properties Reservation and VirtualQuantity are always identical for any resource allocation out of the resource pool.

The host system hosts a virtual system that is represented by the CIM_ComputerSystem instance VS. The hosted relationship is shown through a CIM_HostedDependency instance.

The head element of the "state" virtual system configuration is the VSSD instance STA_VSSD; it is associated with VS through a CIM_SettingsDefineState (SDS) instance. The "State" virtual system configuration contains the SASD instance DISK_STA that represents a storage resource allocation assigned to the virtual system. The virtual disk that is the result of the storage resource allocation is represented as part of the virtual system representation by the CIM_LogicalDisk instance VS_DISK.

NOTE All instances in Figure 5 that are marked with light yellow color represent "State" entities that exist only as long as the virtual system is instantiated (that is, in a state other than "Defined"). These instances do not exist while the virtual system is not instantiated (that is, in the "Defined" state).

The head element of the "defined" virtual system configuration is the VSSD instance DEF_VSSD; it is associated with the head element of the "state" virtual system configuration through an instance of the CIM_ElementSettingData association where the value of the IsDefault property is 1 (Is Default) (abbreviated as ESD(D) in Figure 5). The "defined" virtual system configuration contains the SASD instance DISK_DEF that represents the respective storage resource allocation request. When the virtual system is activated, respective storage resources are allocated based on their definition.

Similarly to the representation of the allocation capabilities of a resource pool or system, the mutability of both the storage resource allocation request in the "Defined" virtual system configuration and of the storage resource allocation in the "State" virtual system configuration is represented by CIM_AllocationCapabilities instances with associated SASD instances through parameterized CIM_SettingsDefineCapabilities instances designating the minimum, maximum, and increment for storage resource allocation changes.

Acceptable virtual system states for the removal of virtual disks are different for the storage resource allocation request and the storage resource allocation. The storage resource allocation can be removed only while the virtual system remains instantiated, as indicated by a value of 2 (Enabled) in the CIM_AllocationCapabilities instance STA_MUT. This is a manifestation of the previously mentioned fact that the "state" configuration is not present while the virtual system is in the "defined" state.

9.2 Inspection

This set of use cases describes how to obtain various CIM instances that represent storage-related information of host and virtual systems.

9.2.1 Inspect the set of virtual disks of an active virtual system

Preconditions: All of the following:

- The client knows a reference to the CIM_ComputerSystem instance that represents the active virtual system.

Flow of activities:

1) From the CIM_ComputerSystem instance the client resolves the CIM_SystemDevice association to find the CIM_LogicalDisk instances that represent virtual disks.
2) For each element of the result set of step 1) the client applies the use case in 9.2.2.

Postconditions: The client knows the virtual disks of the virtual system and their properties.

9.2.2 Inspect the properties of a virtual disk

Preconditions: All of the following:

- The client knows a reference to the instance of the CIM_LogicalDisk class that represents the virtual disk.

Flow of activities:

- The client obtains the CIM_LogicalDisk instance, using the GetInstance( ) intrinsic operation. In that instance, the client interprets property values such as the following:
  – The value of the BlockSize property conveys the block size in effect for the virtual disks
  – The value of the NumberOfBlocks property conveys the size of the virtual disk as seen by the virtual system as a number of blocks
  – The value of the Name property conveys the name of the virtual disk as seen by the virtual system

Postconditions: The client knows properties of the virtual disk.

9.2.3 Determine the allocation capabilities or allocation mutability

This use case is applicable in two cases:

- Case (A) – Determine the capabilities of a system or a resource pool: In this case the entry element is the CIM_System instance representing the host system or the CIM_ResourcePool instance representing the resource pool.
- Case (B) – Determine the mutability of a resource allocation request or resource allocation: In this case the entry element is the RASD or SASD instance representing the resource allocation request or the resource allocation.

Preconditions: The client knows the instance path of the entry element.

Flow of activities:

1) The client invokes the AssociatorsNames( ) intrinsic operation from the entry element through the CIM_ElementCapabilities association to obtain the set of instance paths to those CIM_AllocationCapabilities instances that represent the allocation capabilities (case (A)) or mutability (case (B)) of the entry element.
2) For each instance path obtained in step 1) the client invokes the References( ) intrinsic operation to obtain the set of instances of the CIM_SettingsDefineCapabilities association that reference each CIM_AllocationCapabilities instance from step 1).
3) For each CIM_SettingsDefineCapabilities instance obtained in step 2) the client inspects the values of the ValueRole and ValueRange properties; these values define the type of limitation imposed by the RASD instance that is referenced by the value of the PartComponent property in the CIM_SettingsDefineCapabilities association instance, as follows:
   – A default setting is designated through a value of 0 (Default) for the ValueRole property and a value of 0 (Point) for the ValueRange property. A default setting does not apply for the description of mutability.
   – A minimum setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 1 (Minimums) for the ValueRange property.
   – A maximum setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 2 (Maximums) for the ValueRange property.
   – An increment setting is designated through a value of 3 (Supported) for the ValueRole property and a value of 3 (Increments) for the ValueRange property.
4) For each of the CIM_SettingsDefineCapabilities association instances obtained in step 2) and inspected in step 3) the client invokes the intrinsic GetInstance( ) CIM operation, using the value of the PartComponent property as input for the InstanceName parameter. The result each time is a RASD instance where values of all non-null numeric properties describe the settings in the context established by the CIM_SettingsDefineState instance inspected in step 3).

Postconditions: The client knows the allocation capabilities of the system or the resource pool (case (A)), or the mutability of a resource allocation request or a resource allocation (case (B)).

9.2.4 Determine the default resource allocation capabilities

Preconditions: The client knows all of the following:

- A reference to the CIM_System instance that represents the host system.
- A selected resource type (such as for example 31 (Logical Disk) or 17 (Disk Drive))

Flow of activities:

1) The client obtains instances of the CIM_ElementCapabilities association that reference the instance of the CIM_System class, invoking the References( ) intrinsic operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the instance of the CIM_System class.

   – The value of the ResultClass parameter is set to "CIM_ElementCapabilities".

   The result of step 1) is a set of instances of the CIM_ElementCapabilities association.

2) From the result set of step 1), the client drops those instances where the value set of the Characteristics[ ] array property does not contain an element with the value 2 (Default).

   The result of this step is a set of instances of the CIM_ElementCapabilities association that reference CIM_AllocationCapabilities instances that represent the default allocation capabilities of the system for a number of resource types.

3) For each of the association instances obtained in step 2), the client obtains the CIM_AllocationCapabilities instance that is referenced by the value of the Capabilities property in the respective association instance, invoking the intrinsic GetInstance( ) CIM operation with the value of the InstanceName parameter set to the value of the Capabilities property.

   The result of step 3) is a set of CIM_AllocationCapabilities instances that represent the system's default allocation capabilities for a number of resource types.

4) From the result set of step 3), the client drops those instances where the value set of the ResourceType property does not match the selected resource type.

   The result of this step is one RASD instance that represents the system's default allocation capabilities for the selected resource type. The client continues as in use case 9.2.3 step 2) in order to determine the set of RASD instances that represent the settings for the default resource allocation capabilities for the selected resource type.

Postconditions: The client knows the default allocation capabilities of the system for the selected resource type.

In the example CIM representation shown in Figure 5, the default allocation capabilities for the storage extent resource type of the system are represented by the CIM_AllocationCapabilities instance CAP and related RASD instances.

9.2.5 Determine the default resource pool

Preconditions: The client knows a reference to the CIM_AllocationCapabilities instance that represents the default resource allocation capabilities of the system for a selected resource type; see 9.2.4.

Flow of activities:

1) The client obtains instances of the CIM_ElementCapabilities association that reference the instance of the CIM_AllocationCapabilities class, invoking the intrinsic References( ) CIM operation with parameter values set as follows:

   – The value of the ObjectName parameter refers to the CIM_AllocationCapabilities instance.

   – The value of the ResultClass parameter is set to “CIM_ElementCapabilities”.

   The result of this step is a set of instances of the CIM_ElementCapabilities association.

2) From the result set of step 1), the client drops those instances where the value set of the Characteristics[ ] array property does not contain an element with the value 2 (Default).

The result of this step is a set of two instances of the CIM_ElementCapabilities association. One association instance references the CIM_ResourcePool instances that represent the default resource pool, and one instance references the CIM_System instance that represents the host system.

3) The client selects the instance of the CIM_ElementCapabilities association from the result of step 2) that references the CIM_ResourcePool instance by comparing the value of the ManagedElement property against the known reference to the CIM_System instance that represents the host system and dropping that association instance. The client uses the remaining association instance from the result set of step 2) to obtain the CIM_ResourcePool instance that is referenced by the value of the ManagedElement property in that association instance, invoking the intrinsic GetInstance( ) CIM operation with the value of the InstanceName parameter set to the value of the ManagedElement property.

   The result of this step is the CIM_ResourcePool instance that represents the system's default resource pool for the selected resource type.

Postconditions: The client knows the default resource pool of the system for the selected resource type.

In the example CIM representation shown in Figure 5, the default storage resource pool is represented by the CIM_ResourcePool instance STOR_POOL.

9.2.6 Obtain the storage resource pool with the largest unreserved capacity

Preconditions:

- The client knows a reference to the CIM_System instance that represents the host system.

Flow of activities:

1) The client resolves the CIM_HostedPool association to find the CIM_ResourcePool instances that represent resource pools hosted by the host system, invoking the intrinsic AssociatorNames( ) CIM operation with parameter values set as follows:
   – The value of the ObjectName parameter refers to the CIM_System instance that represents the host.
   – The value of the AssocClass parameter is set to "CIM_HostedPool".
   – The value of the ResultClass parameter is set to "CIM_ResourcePool".

   The result of this step is a set of CIM_ResourcePool instances that represent resource pools hosted by the host system.

2) The client selects from the result set of step 1) only those instances where the value of the ResourceType property matches 31 (Logical Disk).

   The result is a set of CIM_ResourcePool instances that represent storage resource pools hosted by the host system.

3) The client inspects the value of the Capacity and the Reserved properties in all instances selected with step 2), and each time calculates the amount of unreserved storage capacity by subtracting the value of the Reserved property from the value of the Capacity property.

4) From all pools inspected in step 3), the client selects the one that has the largest free capacity.

5) The client checks the resource pool selected in step 4) for architectural limitations as expressed by the pool's capabilities, applying use case 9.2.3.

Postconditions: The client knows the resource pool with the largest unreserved storage capacity.

In the example CIM representation shown in Figure 5, the client initially would know the CIM_System instance HOST that represents the host system. From there, the client would follow the CIM_HostedPool association to locate CIM_ResourcePool instances. Typically the association resolution would yield more than one instance, including instances that represent resource pools of other resource types; consequently the client is required to select only those instances where the value of the ResourceType property matches 31 (Logical Disk). In Figure 5, there is only one CIM_ResourcePool instance in the result set that is named STOR_POOL. From that instance, the client takes the value of the Capacity property and subtracts the value of the Reserved property (2147483648 – 268435456) byte, yielding 1879048192 blocks (or 1792 GB) as the maximum storage capacity presently available from the pool.

9.3 Management

This set of use cases describes how to create new virtual disks, and how to modify existing virtual disks. These management tasks are described in terms of a virtual system management service, as represented by a CIM_VirtualSystemManagementService instance.

9.3.1 Create virtual disk (block based)

Preconditions: All of the following:

- The client knows a reference to the CIM_ComputerSystem instance that represents the virtual system.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The client has performed the use case and knows the default allocation capabilities of the system.
- The size of the new disk is 256 GB (or 268435456 blocks with a size of 1 KB (1024 bytes)).

Flow of activities:

1) The client locally prepares a SASD instance, with properties set as follows:
   - ResourceType: 31 (Logical Disk) // device type as seen by consumer
   - ResourceSubtype: // implementation dependent
   - PoolID: NULL // implies default pool
   - AllocationUnits: "count" // count of blocks; if value is NULL, the effective value // is implied by pool capabilities
   - VirtualQuantity: 268435456 // 256 GB
   - VirtualQuantityBlockSize: 1024 // may be NULL, implied by pool capabilities
   - VirtualQuantityUnits: "count" // count of blocks; if value is NULL, the effective value // is implied by pool capabilities
   - Reservation: NULL // may be NULL if thin provisioning is not requested; // defaults to the size expressed by the value of the // VirtualQuantity property
   - Limit: NULL // defaults to maximum disk size as expressed by the // value of the VirtualQuantity property
   - Address: "/dev/sda1" // optional; if not specified the implementation // assigns an address
   - Values of all other properties are not set (NULL), requesting a default behavior
2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

- AffectedConfiguration: REF to the VSSD instance that represents the "defined" virtual system configuration.
- ResourceSettings: One element with the embedded SASD instance prepared in step 1)

The implementation excutes the AddResourceSettings( ) method

- It is assumed that the method returns 0, indicating successful synchronous execution.
- The initial size of the disk is 256 GB.

Postconditions: A new virtual disk is created for the virtual system, as requested.

9.3.2 Create virtual disk (file based with implicit file creation)

Preconditions: All of the following:

- The client knows a reference to the VSSD instance that represents the virtual system configuration to receive the new virtual disk.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The size of the new disk is 256 GB (or 268435456 KB).
- The initial host file space to reserve is 64 GB (or 67108864 KB).
- The requested name for the file is "FILE1" (relative file path); the files does not exist initially.

Flow of activities:

1) The client locally prepares a SASD instance, with properties set as follows:
   - ResourceType: 31 (Logical Disk)
   - ResourceSubtype: "DMTF:generic:scsi" // SCSI disk
   - PoolID: "FILE_DISK_POOL" // implementation specific // dummy pool
   - AllocationUnits: "count" // number of blocks
   - VirtualQuantity: 268435456 // 256 GB, disk size as seen by virtual system
   - Reservation: 67108864 // 64 GB, initial file size
   - Limit: NULL // defaults to maximum disk size as expressed // by the value of the VirtualQuantity property
   - HostResource[0]: "FILE://FILE1" // client chosen name and location // for the new file
   - Address: "0.7.0.0:0" // SCSI lun 0 on target 0 via bus 0 initiator 7 a // partition 0
   - Values of all other properties are not set (NULL), requesting a default
2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:
   - AffectedConfiguration: REF to the VSSD instance that represents the “defined” virtual system configuration.
   - ResourceSettings: One element with the embedded instance prepared in step 1)

   The implementation excutes the AddResourceSettings( ) method.

– It is assumed that the method returns 0, indicating successful synchronous execution.

– As a side effect, the new file FILE1 is created in a default directory. The initial size of the file is 64 GB, up to a limit of 256 GB. The disk size as seen by the virtual system is 256 GB from the beginning.

Postconditions: A new file-based virtual disk is created for the virtual system, as requested.

Figure 6 shows the situation that results after the create disk operation completed and the virtual system was activated:

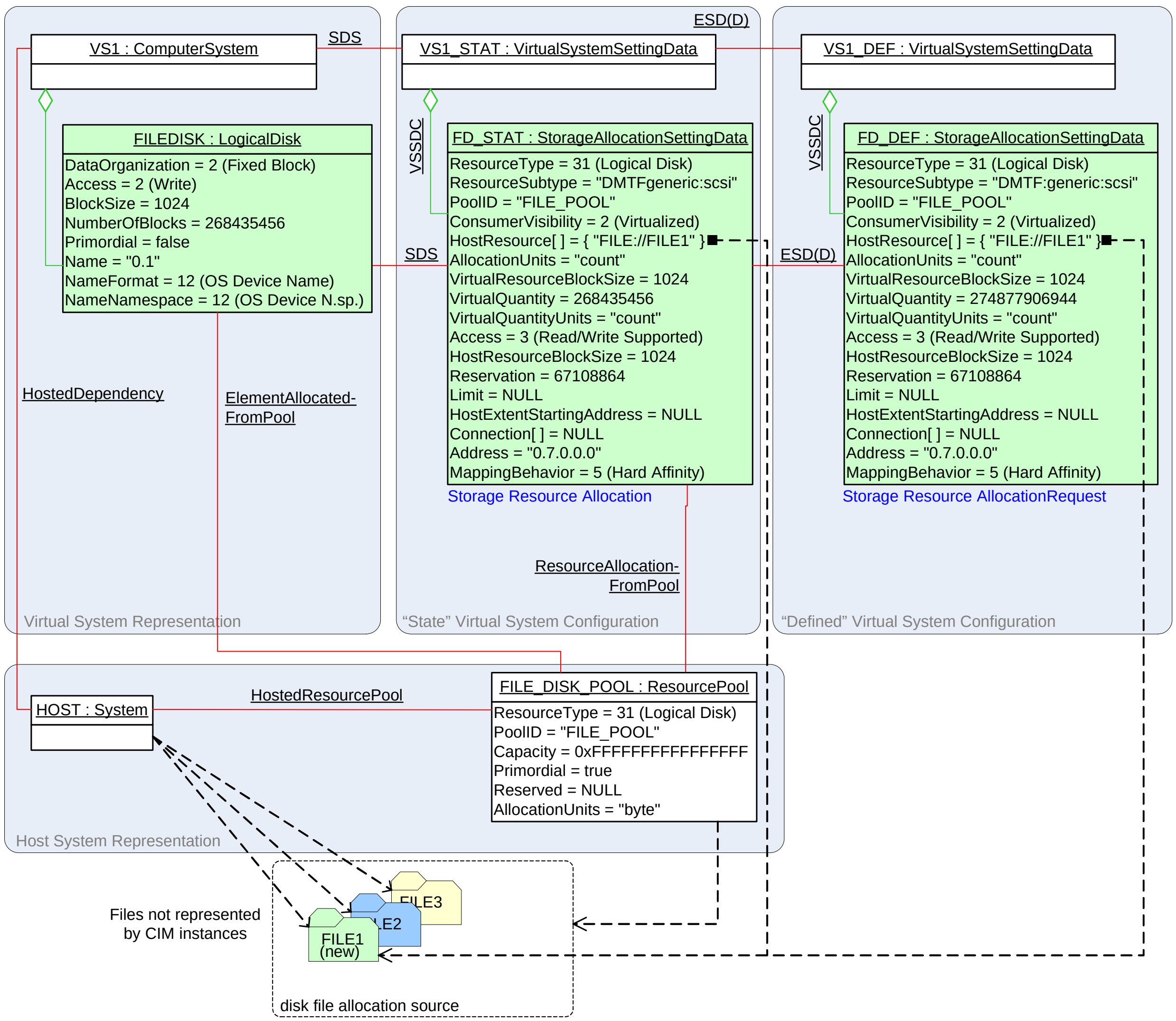

Figure 6 – Create virtual disk with implicit file creation

9.3.3 Create virtual disk (file based pre-existing)

Preconditions: All of the following:

- The client knows a reference to the VSSD instance that represents the virtual system configuration to receive the new virtual disk.

- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The client knows the URI "FILE:://FILE2" of a pre-existing file that contains the data for the new disk.
- The size of the new disk is implied by the content stored in the file.

Flow of activities:

1) The client locally prepares a SASD instance, with properties set as follows:
   - ResourceType: 31 (Logical Disk)
   - ResourceSubtype: "DMTF:generic:scsi" // Microsoft SCSI disk
   - PoolID: "FILE_DISK_POOL" // implementation specific // dummy pool
   - AllocationUnits: "count" // number of blocks
   - VirtualQuantity: NULL // disk size implied by file
   - Reservation: NULL // disk reservation implied by file
   - HostResource[0]: "FILE://FILE2" // URI refering to pre-existing file
   - Address: "0.7.0.0.0" // SCSI bus 0 / initiator 7 / target 0 / lun 0 // partition 0
   - MappingBehavior: 5 (Hard Affinity) // implies that the virtual system // requires this disk at startup
   - Values of all other properties are not set (NULL), requesting a default
2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:
   - AffectedConfiguration: REF to the VSSD instance that represents the “Defined” virtual system configuration.
   - ResourceSettings: One element with the embedded instance prepared in step 1)

   The implementation executes the AddResourceSettings( ) method.
   - It is assumed that the method returns 0, indicating successful synchronous execution.
   - The new disk is configured into the virtual system configuration. It is based on the file provided as input. The initial disk size is 268435456 KB, as implied by the disk content stored in the file.

Postconditions: A new file-based virtual disk is created for the virtual system, as requested.

Figure 7 shows the situation that results after the create disk operation completed and the virtual system was activated:

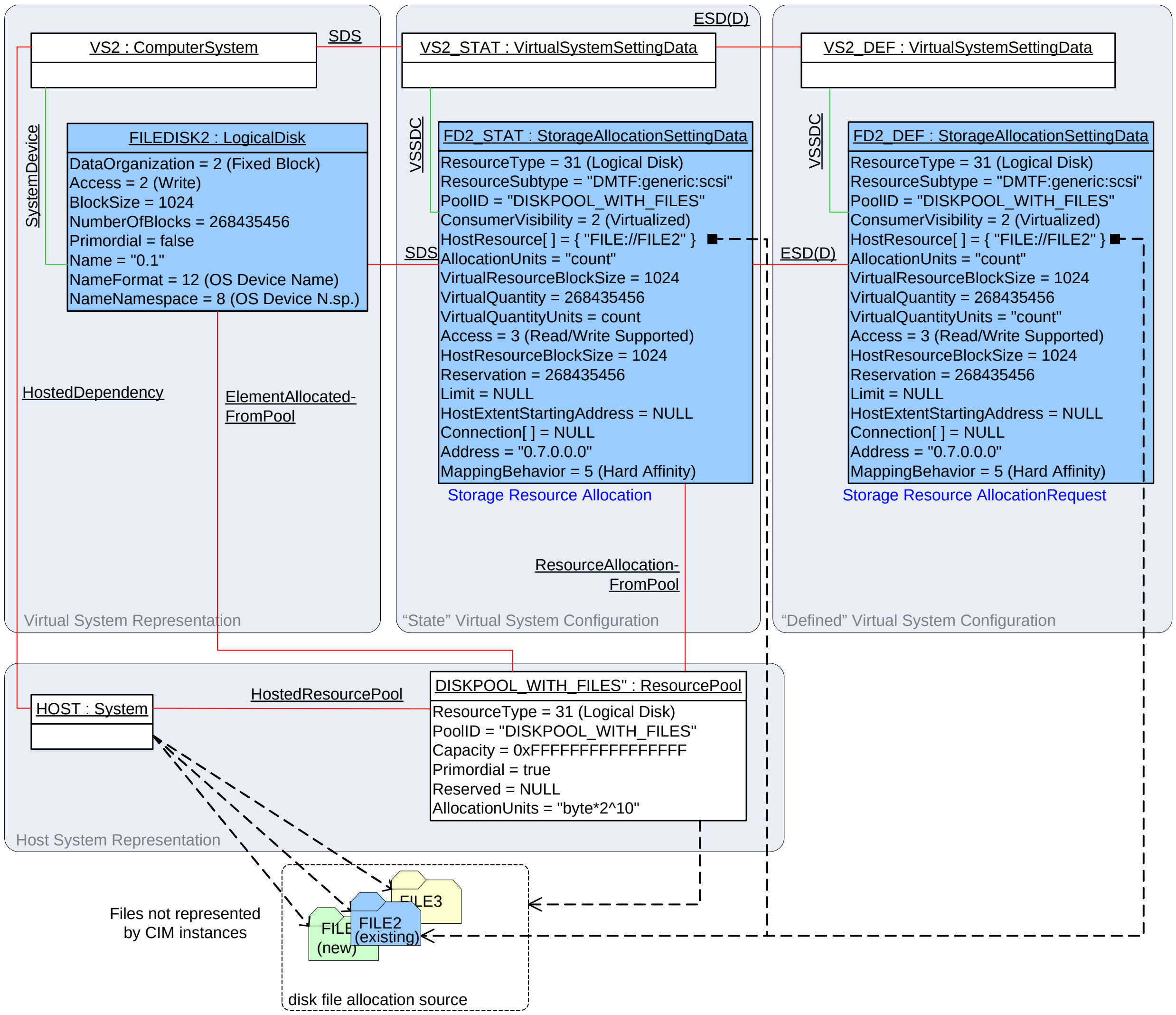

Figure 7 – Create virtual disk with pre-existing file

9.3.4 Create virtual disk (block based passed-through)

Preconditions: All of the following:

- The client knows a reference to the VSSD instance that represents the virtual system configuration to receive the new dedicated virtual disk.
- The client knows a reference to the CIM_VirtualSystemManagementService instance that represents the virtual system management service responsible for the virtual system.
- The size of the new disk is 256 GB (or 268435456 kbyte).

Flow of activities:

1) The client locally prepares an instance of the CIM_ResourceAllocationSettingData class, with properties set as follows:
   - ResourceType: 31 (Logical Disk)
   - ResourceSubtype: "DMTF:xen:vbd"

- PoolID: "PT_POOL" // Example; refers to a pool withdisks // that are passed-through
- AllocationUnits: "count" // number of blocks
- VirtualQuantity: 268435456 // 256 GB; needed if not explicit host resource // requested
- VirtualResourceBlockSize: 1024 // blocksize as seen by consumer
- VirtualQuantityUnits: "count" // number of blocks
- Reservation: 274877906944 // needed if no explicit // host resource requested
- HostResource[0]: "URI(HDISK1)" // optional; may refer to a // specific disk in the pool; if not // specified, the implementation // selects a disk out of the pool
- Address: "/dev/sda1" // optional; if not specified the // implementation assigns an // address
- Values of all other properties are not set (or are NULL), requesting a default

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

   - AffectedConfiguration: REF to CIM_VirtualSystemSettingData instance
   - ResourceSettings: One element with the embedded instance prepared in step 1)

   It is assumed that the method returns 0, indicating successful synchronous execution.

Postconditions: A new passed-through virtual disk is created for the virtual system, as requested.

Figure 8 shows the situation that results after the AddResourceSettings( ) operation completes.

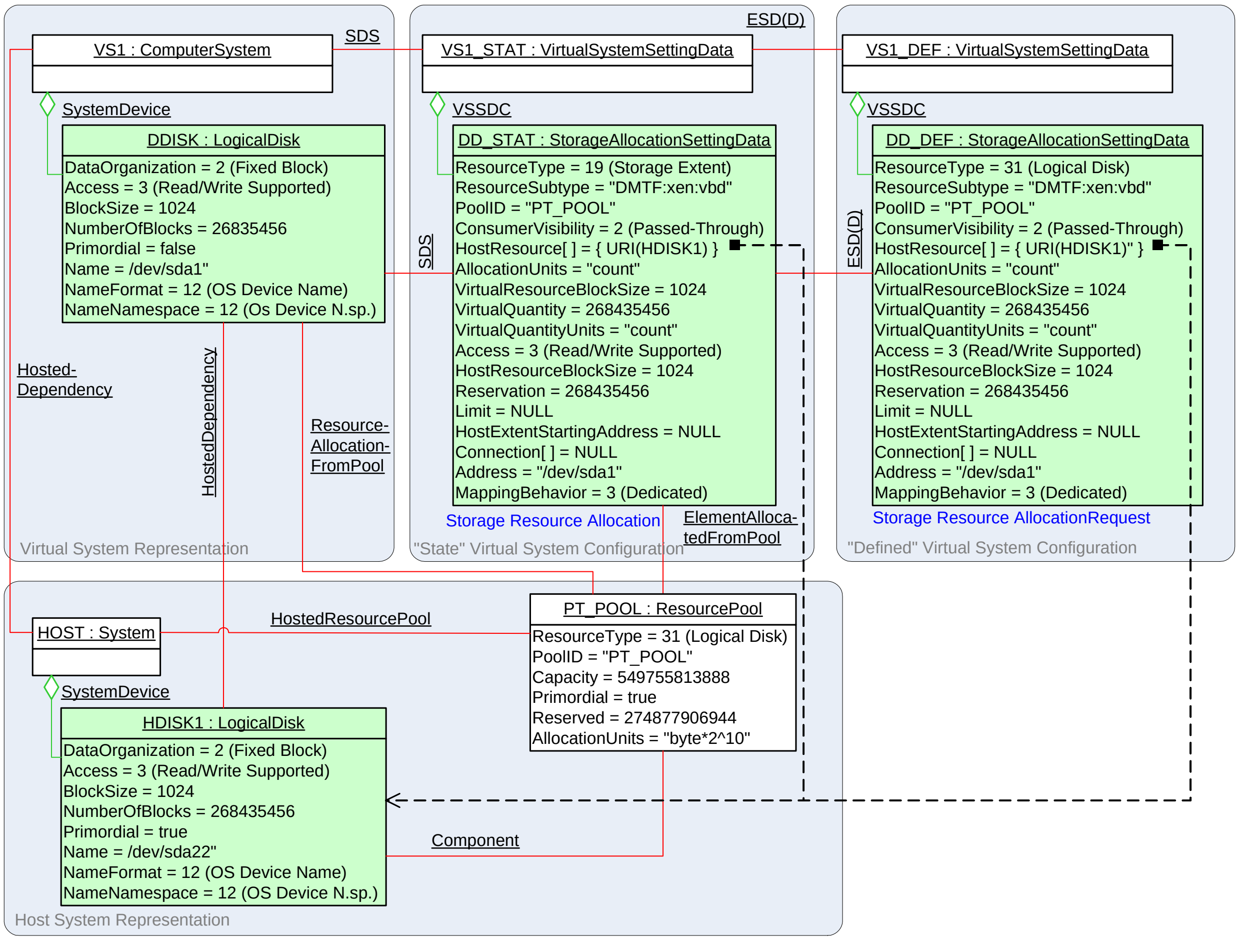

Figure 8 – Create dedicated virtual disk

In this example the resource pool represented by PT_POOL aggregates a set of host disks that were set aside for the purpose of being passed-through to virtual systems. Adding a resource from that pool is actually a selection based upon client requirements.

9.3.5 Create virtual disk (file based delta)

Preconditions: All of the following:

- The situation that was the result of the use case described in 9.3.2.
- The size of the virtual remains disk is 256 GB (or 268435456 KB).
- The initial size of the new delta disk is 16 GB (or 16777216 KB).
- The name of the file is "FILE9" (relative file path); the file does not exist.
- The file is only allocated as the virtual system is activated (instantiated).
- The file is deallocated as the virtual system is deactivated.

Flow of activities:

1) The client locally prepares a SASD instance with properties set as follows:
   – ResourceType: 31 (Logical Disk)

- ResourceSubtype: "DMTF:generic:scsi" // SCSI disk
- PoolID: "FILE_DISK_POOL" // implementation specific // dummy pool
- AllocationUnits: "count" // count of blocks
- VirtualQuantity: 268435456 // 256 GB
- Reservation: 16777216 // 16 GB
- Limit: 67108864 // 64 GB
- AutomaticAllocation: true // fresh extent allocated // on every allocation
- AutomaticDeallocation: true // extent dropped at deallocation time
- HostResource[0]: "FILE://FILE9" // optional; if NULL the // implementation decides
- Parent: URI (FD_DEF) // formatted as specified in DSP0207
- Address: "0.7.0.0.0" // SCSI bus 0 / initiator 7 / target 0 / lun 0 // partition 0
- Values of all other properties are not set (or are NULL), requesting a default

2) The client invokes the AddResourceSettings( ) method of the virtual system management service, with parameters set as follows:

- AffectedConfiguration: REF to VSSD instance // target config
- ResourceSettings: One element with the embedded instance prepared in step 1)

It is assumed that the method that the method returns 0, indicating successful synchronous execution.

Postconditions: A new delta file-based virtual disk is created for the virtual system, as requested.

Figure 9 shows the situation that results after the create delta disk operation completes.

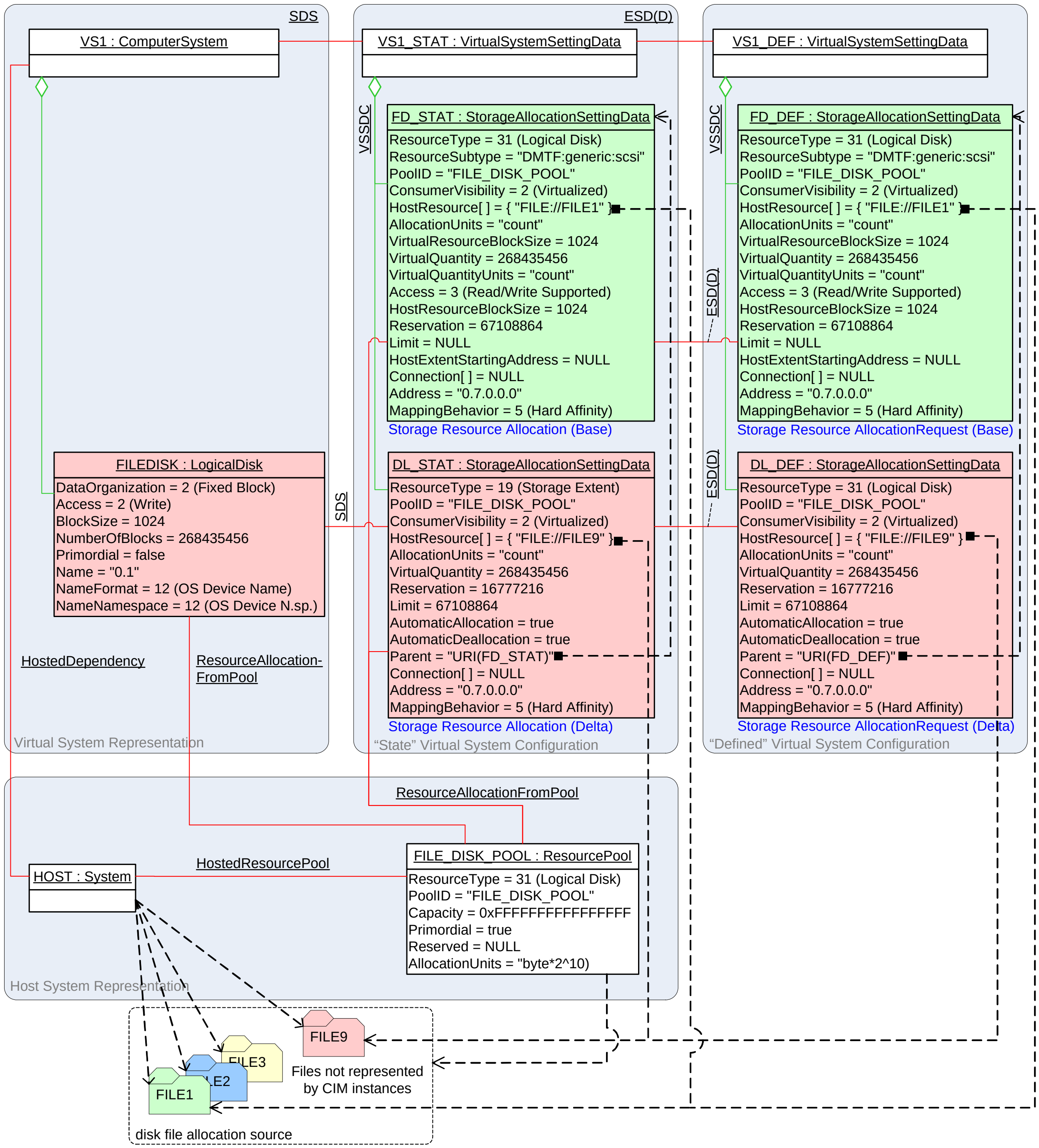

Figure 9 – Create virtual delta disk and file

Note that the instances FD_STAT, DL_STAT and FILEDISK are present only while the virtual system is instantiated and the virtual disk is allocated. Note that there is only one disk FILEDISK in the virtual system representation that is allocated based on both FD_STAT and DL_STAT. There is no separate instance of CIM_LogicalDisk representing each allocation separately as there is only one virtual disk presented to the virtual system.

Note that the file FILE9 containing the delta disk is automatically allocated during virtual disk allocation because the value of the AutomaticAllocation property is true; the file is automatically deallocated during virtual disk deallocation because the value of the AutomaticDeallocation property is true. As a consequence the virtual system at startup time receives a virtual disk that is initially based on FILE1; as the virtual system writes onto the disk the delta is maintained in FILE9. The size of FILE9 is driven by the values of the Reservation and the Limit properties in DL_STAT: The initial file size is 16 GB, up to a limit of 64 GB. As a result the virtual system sees a disk with a size of 256 GB (as indicated by the value of the VirtualQuantity property). That disk is initially based on the read-only file-based extent as allocated by FD_STAT. On top of the read-only extent is a temporary delta read-write extent as allocated by DL_STAT that enables overwriting data up to an amount of 64 GB; the delta extent is discarded when the virtual disk is deallocated, such that the next allocation starts with the initial read-only content again.

10 CIM Elements

Table 5 lists CIM elements that are defined or specialized for this profile. Each CIM element shall be implemented as described in Table 5. The CIM Schema descriptions for any referenced element and its sub-elements apply.

Clauses 7 ("Implementation") and 8 ("Methods") may impose additional requirements on these elements; in particular, clause 7 ("Implementation") may impose requirements for CIM instances.

Table 5 – CIM Elements: Storage Resource Virtualization Profile

Element	Requirement	Description
Classes		
CIM_AffectedJobElement	Optional	See DMTF DSP1041:1.1.
CIM_AllocationCapabilities for capabilities	Mandatory	See DMTF DSP1043:1.0.
CIM_AllocationCapabilities for mutability	Optional	See DMTF DSP1043:1.0.
CIM_Component for resource pool	Conditional	See 10.1.
CIM_ConcreteJob	Optional	See DMTF DSP1041:1.1.
CIM_DiskDrive for host disk drives	Conditional	See 10.2.
CIM_DiskDrive for virtual disk drives	Conditional	See 10.3.
CIM_ElementAllocatedFromPool for allocated virtual resources	Mandatory	See 10.4.
CIM_ElementAllocatedFromPool for resource pool hierarchies	Conditional	See 10.5.
CIM_ElementCapabilities for capabilities	Mandatory	See DMTF DSP1043:1.0.
CIM_ElementCapabilities for mutability	Conditional	See DMTF DSP1043:1.0.
CIM_ElementCapabilities for resource pool	Mandatory	See DMTF DSP1041:1.1.
CIM_ElementSettingData for resource allocation request	Mandatory	See 10.6.
CIM_ElementSettingData for resource pool	Mandatory	See 10.7.
CIM_HostedDependency	Optional	See 10.8.
CIM_HostedResourcePool	Mandatory	See DMTF DSP1041:1.1.
CIM_HostedService	Mandatory	See DMTF DSP1041:1.1.
CIM_LogicalDisk for virtual disk	Conditional	See 10.9.
CIM_ReferencedProfile	Mandatory	See 10.10.
CIM_RegisteredProfile	Mandatory	See 10.11.

Element	Requirement	Description
CIM_ResourceAllocationFromPool	Optional	See DMTF DSP1041:1.1.
CIM_ResourceAllocationSettingData for disk drive allocation information	Conditional	See 10.12.
CIM_ResourcePool	Mandatory	See 10.13.
CIM_ResourcePoolConfigurationCapabilities	Mandatory	See DMTF DSP1041:1.1.
CIM_ResourcePoolConfigurationService	Mandatory	See DMTF DSP1041:1.1.
CIM_SettingsDefineCapabilities	Mandatory	See DMTF DSP1043:1.0.
CIM_SettingsDefineState	Mandatory	See 10.14.
CIM_ServiceAffectsElement	Mandatory	See DMTF DSP1041:1.1.
CIM_StorageAllocationSettingData for storage extent allocation information	Conditional	See 10.15.
CIM_StorageVolume for host storage volume	Conditional	See 10.15.
CIM_StorageExtent for virtual disk	Conditional	See 10.17.
CIM_SystemDevice for host storage volumes	Conditional	See 10.18.
CIM_SystemDevice for virtual resources	Mandatory	See 10.19.
Indications		
None defined		

10.1 CIM_Component for resource pool

The implementation of the CIM_Component association for the representation of the aggregation of host resources into resource pools is conditional.

Condition: The resource aggregation feature (see 7.5) is implemented.

The CIM_Component association is abstract; therefore it cannot be directly implemented. For this reason the provisions in this subclause shall be applied to implementations of subclasses of the CIM_Component association. However, note that clients may directly resolve abstract associations without knowledge of the concrete subclass that is implemented.

Table 6 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1.

Table 6 – Association: CIM_Component for resource pool

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the resource pool. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the CIM_ManagedElement instance that represents a component of the resource pool. **Cardinality:** *

10.2 CIM_DiskDrive for host disk drives

The implementation of the CIM_DiskDrive class for the representation of host disk drives is conditional.

Condition: The resource aggregation feature is implemented for disk drive resource pools; see 7.5.

Table 7 lists the requirements for elements of this class.

Table 7 – Class: CIM_DiskDrive (Host)

Elements	Requirement	Notes
DefaultBlockSize	Mandatory	See CIM schema description

10.3 CIM_DiskDrive for virtual disk drives

The implementation of the CIM_DiskDrive class for the representation of virtual disk drives is conditional.

Condition: This profile is implemented for the allocation of disk drives; see 7.2.

Table 8 lists the requirements for elements of this class.

Table 8 – Class: CIM_DiskDrive (Virtual System)

Elements	Requirement	Notes
EnabledState	Mandatory	Value shall match { 2 | 3 } ("Enabled" | "Disabled").
RequestedState	Optional	Value shall match { 2 | 3 } ("Enabled" | "Disabled").
DefaultBlockSize	Mandatory	See CIM schema description

10.4 CIM_ElementAllocatedFromPool for allocated virtual resources

Table 9 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1.

Table 9 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the instance of a CIM_LogicalDevice subclass that represents the allocated device. **Cardinality:** *

10.5 CIM_ElementAllocatedFromPool for resource pool hierarchies

The implementation of the CIM_ElementAllocatedFromPool association for the representation of resource pool hierarchies is conditional.

Condition: The resource pool management feature (see 7.7) is implemented.

Table 10 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1.

Table 10 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the parent resource pool. **Cardinality:** 1
Dependent	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents the child resource pool. **Cardinality:** *

10.6 CIM_ElementSettingData for resource allocation request

Table 11 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1.

Table 11 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the RASD instance that represents the resource allocation. **Cardinality:** 1
SettingData	Mandatory	**Key:** Value shall reference the RASD instance that represents corresponding the resource allocation request. **Cardinality:** 1
IsDefault	Mandatory	Value shall be 1 (Is Default).

10.7 CIM_ElementSettingData for resource pool

Table 12 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1.

Table 12 – Association: CIM_ElementSettingData

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the CIM_ResourcePool instance that represents a child resource pool. **Cardinality:** 1

Elements	Requirement	Notes
SettingData	Mandatory	**Key:** Value shall reference the RASD instance that represents corresponding the resource allocation request. **Cardinality:** 1

10.8 CIM_HostedDependency

The implementation of the CIM_HostedDependency association is optional.

Table 13 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1 (*Resource Allocation Profile*).

Table 13 – Association: CIM_HostedDependency

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the instance of the CIM_LogicalDevice class that represents a dedicated host device. **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Value shall reference the instance of the CIM_LogicalDevice class that represents a virtual device. **Cardinality:** 0..1

10.9 CIM_LogicalDisk for virtual disk

The implementation of the CIM_LogicalDisk class for the representation of virtual disks is conditional.

Condition: This profile is implemented for the allocation of storage extents; see 7.2.

Table 14 lists the requirements for elements of this class in addition to those specified for the implementation of the CIM_StorageExtent class for the representation of virtual disks; see 10.17.

Table 14 – Class: CIM_LogicalDisk (Virtual System)

Elements	Requirement	Notes
Name	Mandatory	See 7.9.2.3.
NameFormat	Mandatory	See 7.9.2.4.
NameNamespace	Mandatory	See 7.9.2.5.

10.10 CIM_ReferencedProfile

Table 15 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1033:1.0 (*Profile Registration Profile*).

Table 15 – Association: CIM_ReferencedProfile

Elements	Requirement	Notes

Elements	Requirement	Notes
Antecedent	Mandatory	**Key:** Value shall reference the CIM_RegisteredProfile instance that represents an implementation of this profile. **Cardinality:** 0..1
Dependent	Mandatory	**Key:** Value shall reference the CIM_RegisteredProfile instance that represents an implementation of the scoping profile. **Cardinality:** 0..1

10.11 CIM_RegisteredProfile

Table 16 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM schema and in DMTF DSP1033:1.0 (*Profile Registration Profile*).

Table 16 – Class: CIM_RegisteredProfile

Elements	Requirement	Notes
RegisteredOrganization	Mandatory	Value shall be 2 (DMTF).
RegisteredName	Mandatory	Value shall be "Storage Resource Virtualization".
RegisteredVersion	Mandatory	Value shall be "1.0.0".

10.12 CIM_ResourceAllocationSettingData for disk drive allocation information

The implementation of the CIM_ResourceAllocationSettingData class for the representation of disk drive allocation information is conditional.

Condition: This profile is implemented for the allocation of disk drives; see 7.2.

Table 17 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1 (*Resource Allocation Profile*).

Table 17 – Class: CIM_ResourceAllocationSettingData

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**.
ResourceType	Mandatory	See 7.8.3.1.
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 7.8.3.2.
PoolID	Mandatory	See 7.8.3.3.
ConsumerVisibility	Optional	See 7.8.3.4.
HostResource[ ]	Conditional	See 7.8.3.5.
AllocationUnits	Mandatory	See 7.8.3.6.
VirtualQuantity	Mandatory	See 7.8.3.7.

Elements	Requirement	Notes
VirtualQuantityUnits	Mandatory	EXPERIMENTAL; See 7.8.3.8.
Reservation	Optional	See 7.8.3.9.
Limit	Optional	See 7.8.3.10.
Weight	Optional	See 7.8.3.11.
AutomaticAllocation	Optional	See DMTF DSP1041:1.1.
AutomaticDeallocation	Optional	See DMTF DSP1041:1.1.
Parent	Optional	See 7.8.3.12.
Connection[ ]	Optional	See 7.8.3.13.
Address	Optional	See 7.8.3.14.
MappingBehavior	Optional	See 7.8.3.15.

10.13 CIM_ResourcePool

Table 18 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1 (*Resource Allocation Profile*).

Table 18 – Class: CIM_ResourcePool

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**
PoolID	Mandatory	See 7.4.4.
Primordial	Mandatory	See 7.4.5.
Capacity	Conditional	See 7.4.7.
Reserved	Optional	See 7.4.6.
ResourceType	Mandatory	See 7.4.2.
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 7.4.3.
AllocationUnits	Mandatory	See 7.4.8.
MaxConsumableResource	Optional	See 7.4.9.
CurrentlyConsumedResource	Optional	See 7.4.10.
ConsumedResourceUnit	Optional	See 7.4.11.

10.14 CIM_SettingsDefineState

Table 19 lists the requirements for elements of this association. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1 (*Resource Allocation Profile*).

Table 19 – Association: CIM_SettingsDefineState

Elements	Requirement	Notes
ManagedElement	Mandatory	**Key:** Value shall reference the CIM_ManagedSystemElement instance representing the allocated virtual resource. **Cardinality:** 0..1
SettingData	Mandatory	**Key:** Value shall reference the CIM_ResourceAllocationSettingData instance representing the resource allocation. **Cardinality:** 0..1

10.15 CIM_StorageAllocationSettingData for storage allocation information

The implementation of the CIM_StorageAllocationSettingData class for the representation of storage allocation information is conditional.

Condition: This profile is implemented for the allocation of storage extents; see 7.2.

Table 20 lists the requirements for elements of this class. These requirements are in addition to those specified in the CIM Schema and in DMTF DSP1041:1.1 (*Resource Allocation Profile*).

Table 20 – Class: CIM_StorageAllocationSettingData

Elements	Requirement	Notes
InstanceID	Mandatory	**Key**.
ResourceType	Mandatory	See 7.8.4.1.
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 7.8.3.2.
PoolID	Mandatory	See 7.8.3.3.
ConsumerVisibility	Optional	See 7.8.3.4.
HostResource[ ]	Conditional	See 7.8.4.5.
AllocationUnits	Mandatory	See 7.8.4.6.
VirtualQuantity	Optional for Q_SASD Mandatory otherwise	See 7.8.4.7.
VirtualQuantityUnit	Mandatory	See 7.8.4.8.
Reservation	Optional	See 7.8.4.9.
Limit	Optional	See 7.8.4.10.
Weight	Optional	See 7.8.3.11.
AutomaticAllocation	Optional	See DMTF DSP1041:1.1.
AutomaticDeallocation	Optional	See DMTF DSP1041:1.1.
Parent	Optional	See 7.8.3.12.
Connection[ ]	Optional	See 7.8.3.13.
Address	Optional	See 7.8.3.14.
MappingBehavior	Optional	See 7.8.3.15.
VirtualResourceBlockSize	Mandatory	See 7.8.4.16.
Access	Optional	See 7.8.4.17.
HostResourceBlockSize	Mandatory	See 7.8.4.18.

Elements	Requirement	Notes
HostExtentStartingAddress	Optional	See 7.8.4.19.
HostExtentName	Optional	See 7.8.4.20.
HostExtentNameFormat	Conditional	See 7.8.4.21.
OtherHostExtentNameFormat	Conditional	See 7.8.4.22.
HostExtentNameNamespace	Conditional	See 7.8.4.23.
OtherHostExtentNameNamespace	Conditional	See 7.8.4.24.

10.16 CIM_StorageVolume for host storage volume

The implementation of the CIM_StorageVolume class for the representation of host storage volumes is conditional.

Condition: The storage resource aggregation feature is implemented; see 7.5.

Table 21 lists the requirements for elements of this class.

Table 21 – Class: CIM_StorageVolume for host storage volume

Elements	Requirement	Notes
Access	Mandatory	See CIM Schema description.
BlockSize	Mandatory	See CIM Schema description.
NumberOfBlocks	Mandatory	See CIM Schema description.
Name	Mandatory	See CIM Schema description.
NameFormat	Mandatory	See CIM Schema description.
NameNamespace	Mandatory	See CIM Schema description.

10.17 CIM_StorageExtent for virtual storage extent

See 7.9 for detailed implementation requirements for this class if it is used for the representation of virtual disks.

Table 22 lists the requirements for elements of this class.

Table 22 – Class: CIM_StorageExtent for virtual disks

Elements	Requirement	Notes
BlockSize	Mandatory	See 7.9.2.1.
NumberOfBlocks	Mandatory	Value shall reflect the number of blocks available to the virtual system.
Name	Mandatory	Value may reflect the name of the virtual disk.
NameFormat	Mandatory	See CIM schema description.
NameNamespace	Mandatory	See CIM schema description.

10.18 CIM_SystemDevice for host storage volumes

The implementation of the CIM_SystemDevice association for host storage volumes is conditional.

Condition: The storage resource aggregation feature is implemented; see 7.5.

Table 23 lists the requirements for elements of this association.

Table 23 – Association: CIM_SystemDevice for host storage volumes

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_System class. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_StorageVolume class. **Cardinality:** *

10.19 CIM_SystemDevice for virtual resources

Table 24 lists the requirements for elements of this association.

Table 24 – Association: CIM_SystemDevice for virtual resources

Elements	Requirement	Notes
GroupComponent	Mandatory	**Key:** Value shall reference an instance of the CIM_ComputerSystem class representing the virtual system. **Cardinality:** 1
PartComponent	Mandatory	**Key:** Value shall reference the instance of the CIM_LogicalDisk, CIM_StorageVolume, CIM_StorageExtent or CIM_DiskDrive class representing the virtual resource. **Cardinality:** *

ANNEX A
(Informative)

Change Log

Version	Date	Description
1.0.0	2010-04-22	DMTF Standard Release