

1



2

Document Number: DSP1047

3

Date: 2010-04-22

4

5

Version: 1.0.0

6

Storage Resource Virtualization Profile

7

Document Type: Specification

8

Document Status: DMTF Standard

9

Document Language: E

10
11
12

Copyright Notice

Copyright © 2010 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

13
14
15
16

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

17
18
19
20
21
22
23
24
25
26
27
28
29

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party's reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

30
31
32

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit <http://www.dmtf.org/about/policies/disclosures.php>.

CONTENTS

34	Foreword	7
35	Introduction	8
36	1 Scope	9
37	2 Normative references	9
38	3 Terms and definitions	10
39	4 Symbols and abbreviated terms	13
40	5 Synopsis	14
41	6 Description	15
42	6.1 General	15
43	6.2 Storage resource virtualization class schema	16
44	6.3 Resource pools	17
45	6.3.1 General	18
46	6.3.2 Representation of host resources	18
47	6.3.3 Primordial resource pool	19
48	6.3.4 Concrete resource pool	19
49	6.3.5 Hierarchies of resource pools	19
50	6.3.6 Resource pool management	20
51	6.4 Resource allocation	20
52	6.4.1 General	21
53	6.4.2 Storage resource allocations backed by files	21
54	6.4.3 Resource allocation request	24
55	6.4.4 Resource allocation	24
56	6.4.5 Virtual disk	24
57	6.4.6 Virtual disk drive	24
58	6.4.7 Storage virtualization	25
59	6.4.8 Dedicated host storage	25
60	6.4.9 Virtual storage extended configurability	25
61	6.4.10 Management of host storage resources through SMI-S profiles	25
62	7 Implementation	27
63	7.1 Common requirements	27
64	7.2 Resource types	27
65	7.2.1 General	28
66	7.2.2 Logical disks, storage volumes and storage extents	28
67	7.2.3 Disk drives	28
68	7.3 Host resources	28
69	7.3.1 Host storage volume	28
70	7.3.2 Host disk drives	28
71	7.4 Resource pools	29
72	7.4.1 General	29
73	7.4.2 ResourceType property	29
74	7.4.3 ResourceSubType property	29
75	7.4.4 Primordial property	30
76	7.4.5 PoolID property	30
77	7.4.6 Reserved property	30
78	7.4.7 Capacity property	31
79	7.4.8 AllocationUnits property	31
80	7.4.9 MaxConsumableResource property	31
81	7.4.10 CurrentlyConsumedResource property	32
82	7.4.11 ConsumedResourceUnit property	32
83	7.4.12 Instance requirements	32

84	7.5	Resource aggregation feature	32
85	7.6	Resource pool hierarchies feature	32
86	7.7	Resource pool management feature	33
87	7.8	Resource allocation	33
88	7.8.1	General	33
89	7.8.2	Flavors of allocation data	33
90	7.8.3	CIM_ResourceAllocationSettingData properties	34
91	7.8.4	CIM_StorageAllocationSettingData properties	37
92	7.8.5	Instance requirements	41
93	7.9	Virtual resources	42
94	7.9.1	Virtual resource instance requirements	42
95	7.9.2	CIM_StorageExtent properties	43
96	8	Methods	44
97	8.1	Profile conventions for operations	44
98	8.2	CIM_DiskDrive for host disk drives	44
99	8.3	CIM_DiskDrive for virtual disk drives	44
100	8.4	CIM_LogicalDisk for virtual disk drives	44
101	8.5	CIM_ReferencedProfile	45
102	8.6	CIM_RegisteredProfile	45
103	8.7	CIM_StorageAllocationSettingData for storage extent allocation information	45
104	8.8	CIM_StorageExtent for virtual disk	45
105	8.9	CIM_SystemDevice for host storage volumes	45
106	8.10	CIM_SystemDevice for virtual resources	45
107	9	Use cases	45
108	9.1	Instance diagram	45
109	9.2	Inspection	48
110	9.2.1	Inspect the set of virtual disks of an active virtual system	48
111	9.2.2	Inspect the properties of a virtual disk	48
112	9.2.3	Determine the allocation capabilities or allocation mutability	49
113	9.2.4	Determine the default resource allocation capabilities	49
114	9.2.5	Determine the default resource pool	50
115	9.2.6	Obtain the storage resource pool with the largest unreserved capacity	51
116	9.3	Management	52
117	9.3.1	Create virtual disk (block based)	52
118	9.3.2	Create virtual disk (file based with implicit file creation)	53
119	9.3.3	Create virtual disk (file based pre-existing)	54
120	9.3.4	Create virtual disk (block based passed-through)	56
121	9.3.5	Create virtual disk (file based delta)	58
122	10	CIM Elements	61
123	10.1	CIM_Component for resource pool	62
124	10.2	CIM_DiskDrive for host disk drives	63
125	10.3	CIM_DiskDrive for virtual disk drives	63
126	10.4	CIM_ElementAllocatedFromPool for allocated virtual resources	63
127	10.5	CIM_ElementAllocatedFromPool for resource pool hierarchies	63
128	10.6	CIM_ElementSettingData for resource allocation request	64
129	10.7	CIM_ElementSettingData for resource pool	64
130	10.8	CIM_HostedDependency	65
131	10.9	CIM_LogicalDisk for virtual disk	65
132	10.10	CIM_ReferencedProfile	65
133	10.11	CIM_RegisteredProfile	66
134	10.12	CIM_ResourceAllocationSettingData for disk drive allocation information	66
135	10.13	CIM_ResourcePool	67
136	10.14	CIM_SettingsDefineState	67
137	10.15	CIM_StorageAllocationSettingData for storage allocation information	68
138	10.16	CIM_StorageVolume for host storage volume	69
139	10.17	CIM_StorageExtent for virtual storage extent	69

140 10.18 CIM_SystemDevice for host storage volumes..... 70
 141 10.19 CIM_SystemDevice for virtual resources..... 70
 142 ANNEX A (Informative) Change Log 71

143

144 **Figures**

145 Figure 1 – Storage Resource Virtualization Profile: Profile class diagram 16
 146 Figure 2 – Instance diagram: Concept of storage resource pool hierarchies 20
 147 Figure 3 – Instance diagram: Concept of storage resource allocation 23
 148 Figure 4 – Cooperation of DMTF SVPC and SNIA SMI-S profiles 26
 149 Figure 5 – Instance diagram: Example CIM representation of storage resource virtualization 46
 150 Figure 6 – Create virtual disk with implicit file creation 54
 151 Figure 7 – Create virtual disk with pre-existing file 56
 152 Figure 8 – Create dedicated virtual disk 58
 153 Figure 9 – Create virtual delta disk and file..... 60

154

155 **Tables**

156 Table 1 – Related Profiles..... 14
 157 Table 2 – Optional Features..... 15
 158 Table 3 – Predefined ResourceSubType values 30
 159 Table 4 – Acronyms for RASD adapted for the representation of various flavors of allocation data..... 34
 160 Table 5 – CIM Elements: Storage Resource Virtualization Profile 61
 161 Table 6 – Association: CIM_Component for resource pool 62
 162 Table 7 – Class: CIM_DiskDrive (Host) 63
 163 Table 8 – Class: CIM_DiskDrive (Virtual System) 63
 164 Table 9 – Association: CIM_ElementSettingData 63
 165 Table 10 – Association: CIM_ElementSettingData 64
 166 Table 11 – Association: CIM_ElementSettingData 64
 167 Table 12 – Association: CIM_ElementSettingData 64
 168 Table 13 – Association: CIM_HostedDependency 65
 169 Table 14 – Class: CIM_LogicalDisk (Virtual System) 65
 170 Table 15 – Association: CIM_ReferencedProfile 65
 171 Table 16 – Class: CIM_RegisteredProfile 66
 172 Table 17 – Class: CIM_ResourceAllocationSettingData 66
 173 Table 18 – Class: CIM_ResourcePool 67
 174 Table 19 – Association: CIM_SettingsDefineState 68
 175 Table 20 – Class: CIM_StorageAllocationSettingData 68
 176 Table 21 – Class: CIM_StorageVolume for host storage volume 69
 177 Table 22 – Class: CIM_StorageExtent for virtual disks 69
 178 Table 23 – Association: CIM_SystemDevice for host storage volumes 70
 179 Table 24 – Association: CIM_SystemDevice for virtual resources 70

180

182

Foreword

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

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

187 The authors wish to acknowledge the following people.

188 • Editors:

189 – Michael Johanssen – IBM

190 – Mike Walker – IBM

191 • Participants from the DMTF System Virtualization, Partitioning and Clustering Working Group:

192 – Gareth Bestor – IBM

193 – Jim Fehlig – Novell

194 – Michael Johanssen – IBM

195 – Larry Lamers – VMware

196 – Andreas Maier – IBM

197 – John Parchem – Microsoft

198 – Shishir Pardikar – XenSource

199 – Nihar Shah – Microsoft

200 – David Simpson – IBM

201 – Mike Walker – IBM

202

203

Introduction

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

209 Document conventions

210 Typographical conventions

211 The following typographical conventions are used in this document:

- 212 • Document titles are marked in *italics*.

213 Experimental material

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

219 The following typographical convention indicates experimental material:

220 **EXPERIMENTAL**

221 Experimental material appears here.

222 **EXPERIMENTAL**

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

225

226

Storage Resource Virtualization Profile

227 1 Scope

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

230 2 Normative references

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

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

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

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

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

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

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

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

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

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

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

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

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

258 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*
259 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

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

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

265 Clause 6: *Generic Target Ports* profile 1.0

266 Clause 14: *Generic Initiator Ports* profile 1.0

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

268 Clause 5: *Block Services* package 1.3

269 Clause 15: *Extent Composition* subprofile 1.2

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

271 Clause 6: *Storage HBA* profile 1.3

272 Clause 7: *Host Discovered Resources* profile 1.2

273 3 Terms and definitions

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

277 3.1

278 **can**

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

280 3.2

281 **cannot**

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

283 3.3

284 **conditional**

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

287 3.4

288 **mandatory**

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

291 3.5

292 **may**

293 indicates a course of action permissible within the limits of the document

294 3.6

295 **need not**

296 indicates a course of action permissible within the limits of the document

297 3.7

298 **optional**

299 indicates a course of action permissible within the limits of the document

- 300 **3.8**
301 **referencing profile**
302 indicates a profile that owns the definition of this class and can include a reference to this profile in its
303 “Related Profiles” table
- 304 **3.9**
305 **shall**
306 indicates requirements strictly to be followed in order to conform to the document and from which no
307 deviation is permitted
- 308 **3.10**
309 **shall not**
310 indicates requirements strictly to be followed in order to conform to the document and from which no
311 deviation is permitted
- 312 **3.11**
313 **should**
314 indicates that among several possibilities, one is recommended as particularly suitable, without
315 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 316 **3.12**
317 **should not**
318 indicates that a certain possibility or course of action is deprecated but not prohibited
- 319 **3.13**
320 **unspecified**
321 indicates that this profile does not define any constraints for the referenced CIM element
- 322 **3.14**
323 **client**
324 an application that exploits facilities specified by this profile
- 325 **3.15**
326 **implementation**
327 a set of CIM providers that realize the classes specified by this profile
- 328 **3.16**
329 **this profile**
330 this DMTF management profile – the *Storage Resource Virtualization Profile*
- 331 **3.17**
332 **concrete storage resource pool**
333 a storage resource pool that subdivides the capacity of its (primordial or concrete) parent resource pool
- 334 **3.18**
335 **host storage resource**
336 a storage resource that exists in scope of or is accessible by a host system. A host system may contain or
337 have access to one or more storage resources that may be as a whole or partially allocated to virtual
338 systems
- 339 **3.19**
340 **host system**
341 the scoping system that contains or has access to storage resources that may be allocated, virtualized, or
342 both

- 343 **3.20**
344 **initiator port**
345 a port that acts as the source for a data exchange operation
- 346 **3.21**
347 **logical disk**
348 the instantiation of allocated host resources that is exposed to a virtual system through a storage device;
349 the result of a storage resource allocation based on a storage resource allocation request
- 350 **3.22**
351 **port**
352 communication endpoint for systems or storage devices. A port enables the exchange of data according
353 to one or more protocols
- 354 **3.23**
355 **primordial storage resource pool**
356 a storage resource pool that aggregates storage resources available for or used by storage resource
357 allocations
- 358 **3.24**
359 **storage pool**
360 a special kind of storage resource pool that is managed through SMI-S
- 361 **3.25**
362 **storage resource**
363 a logical disk, a storage volume or a storage extent
- 364 **3.26**
365 **storage resource allocation**
366 the allocation of a storage resource from a storage resource pool to a virtual system
- 367 **3.27**
368 **storage resource allocation request**
369 a request for a storage resource allocation
- 370 **3.28**
371 **storage resource pool**
372 a resource pool that represents storage resources available for storage resource allocation
- 373 **3.29**
374 **storage resource pool configuration service**
375 a configuration service that supports the addition or removal of host storage resources to or from a
376 storage resource pool, and the creation or deletion of concrete subpools of a storage resource pool
- 377 **3.30**
378 **storage volume**
379 the instantiation of allocated host resources that is exposed to a virtual system through a storage device
380 that is published for use outside of the scoping system. Like a logical disk, a storage volume is the result
381 of a storage resource allocation based on a storage resource allocation request
- 382 **3.31**
383 **target port**
384 a port that acts as a target of a data exchange operation

385 **3.32**

386 **virtual computer system**

387 **virtual system**

388 the concept of virtualization as applied to a computer system

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

391 **3.33**

392 **virtualization platform**

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

395 **4 Symbols and abbreviated terms**

396 The following symbols and abbreviations are used in this document.

397 **4.1**

398 **CIM**

399 Common Information Model

400 **4.2**

401 **CIMOM**

402 CIM object manager

403 **4.3**

404 **ESD**

405 CIM_ElementSettingData

406 **4.4**

407 **HBA**

408 host bus adapter

409 **4.5**

410 **RASD**

411 CIM_ResourceAllocationSettingData

412 **4.6**

413 **SASD**

414 CIM_StorageAllocationSettingData

415 **4.7**

416 **SAN**

417 storage area network

418 **4.8**

419 **SDS**

420 CIM_SettingsDefineState

421 **4.9**

422 **SLP**

423 Service Location Protocol

- 424 **4.10**
 425 **SMI-S**
 426 Storage Management Initiative Specification
- 427 **4.11**
 428 **SNIA**
 429 Storage Networking Industry Association
- 430 **4.12**
 431 **VS**
 432 virtual system
- 433 **4.13**
 434 **VSSD**
 435 CIM_VirtualSystemSettingData
- 436 **4.14**
 437 **VSSDC**
 438 CIM_VirtualSystemSettingDataComponent

439 **5 Synopsis**

440 **Profile Name:** Storage Resource Virtualization

441 **Version:** 1.0.0

442 **Organization:** DMTF

443 **CIM Schema Version:** 2.21

444 **Central Class:** CIM_ResourcePool

445 **Scoping Class:** CIM_System

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

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

449

Table 1 – Related Profiles

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

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

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

451 **Table 2 – Optional Features**

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

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

455 6 Description

456 This clause contains informative text only.

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

459 6.1 General

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

462 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:

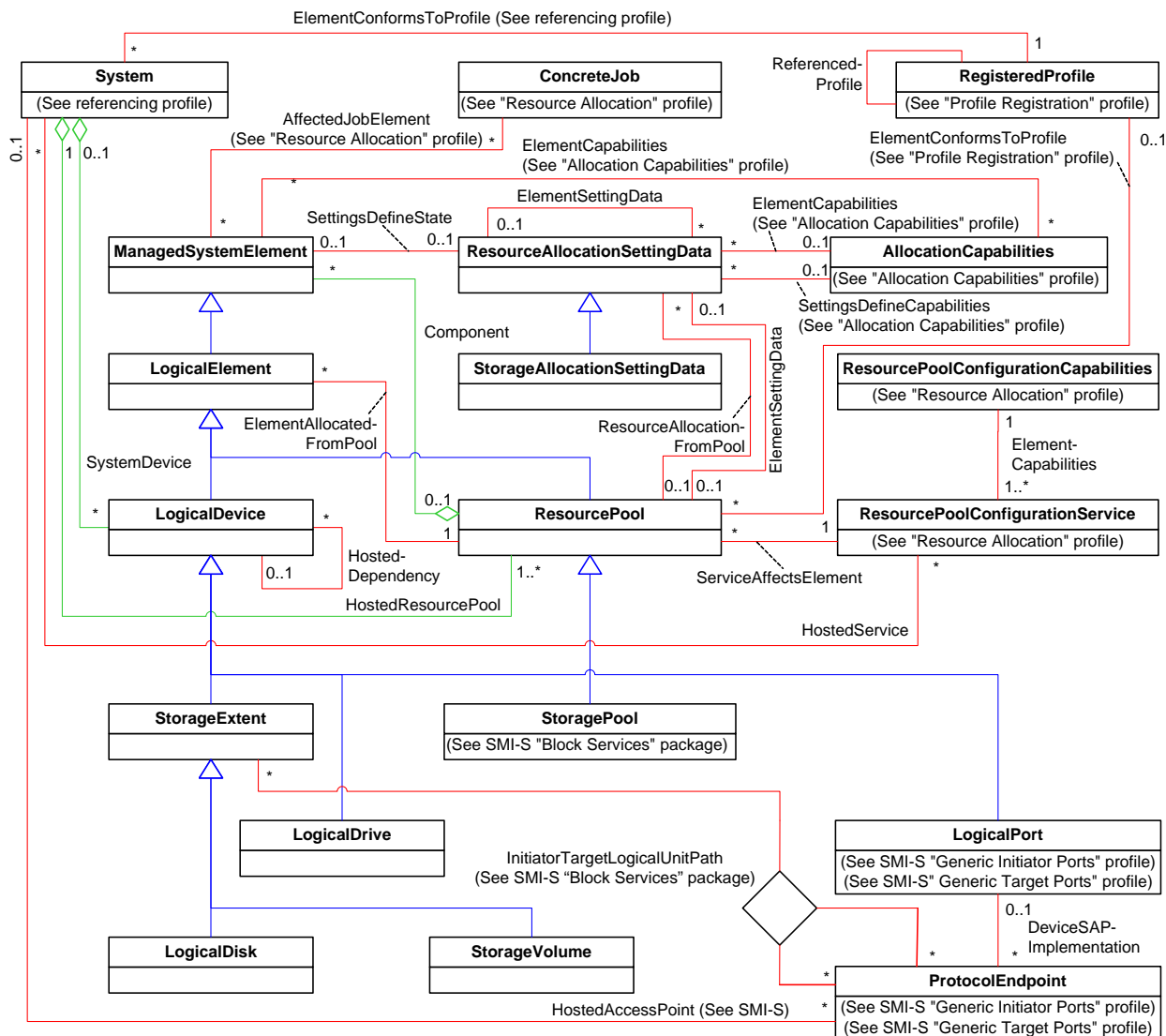
- 465 • virtual storage resources, designated by one of the resource type values 31 (Logical Disk),
 466 32 (Storage Volume) or 19 (Storage Extent)
- 467 • virtual disk drives, designates by the value 17 (Disk Drive)

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

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

474 **6.2 Storage resource virtualization class schema**

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



480

481 **Figure 1 – Storage Resource Virtualization Profile: Profile class diagram**

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

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

- 487 • The CIM_ElementAllocatedFromPool association models hierarchies of resource pools and
488 models the relationship of resource pools and storage resources allocated from those.
- 489 • The CIM_HostedResourcePool association models the hosting dependency between a
490 resource pool and its host system. A host system supports at least one resource pool for
491 storage resources.
- 492 • The CIM_LogicalDisk class, the CIM_StorageVolume class and the CIM_StorageExtent class
493 model the following aspects of logical disks and storage volumes:
 - 494 – logical disks, storage volumes or plain storage extents as devices in the scope of a system,
495 as modeled by the CIM_SystemDevice association
 - 496 – host storage extents (including subtypes) as components within storage resource pools, as
497 modeled by the CIM_Component association
 - 498 – virtual disks as a result of a storage resource allocation from a storage resource pool, as
499 modeled by the CIM_ElementAllocatedFromPool association
- 500 • The CIM_DiskDrive class models the following aspects of disk drives:
 - 501 – disk drives as devices in the scope of a system, as modeled by the CIM_SystemDevice
502 association
 - 503 – disk drives as components within disk drive resource pools, as modeled through the
504 CIM_Component association
 - 505 – disk drives as a result of a disk drive allocation from a disk drive resource pool, as modeled
506 by the CIM_ElementAllocatedFromPool association
- 507 • The CIM_ResourceAllocationSettingData class models disk drive resource allocations or disk
508 drive resource allocation requests
- 509 • The CIM_StorageAllocationSettingData class models storage resource allocations or storage
510 resource allocation requests
- 511 • The CIM_AllocationCapabilities class and the CIM_ElementCapabilities association models
 - 512 – the resource allocation capabilities of host systems
 - 513 – the resource allocation capabilities of storage resource pools
 - 514 – the mutability of existing resource allocations
- 515 • The CIM_SettingsDefineCapabilities association models the relation between allocation
516 capabilities and the settings that define these capabilities
- 517 • The CIM_ResourcePoolConfigurationService class models configuration services for resource
518 pools and the CIM_ResourcePoolConfigurationCapabilities class modeling their capabilities
- 519 • The CIM_ConcreteJob class and the CIM_AffectedJobElement association models
520 asynchronous management tasks initiated through resource pool configuration services
- 521 • The CIM_HostedDependency association models
 - 522 – the relationship between virtual storage extents and host storage extents
 - 523 – the relationship between virtual disk drives and host disk drives

524 **6.3 Resource pools**

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

526 6.3.1 General

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

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

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

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

553 6.3.2 Representation of host resources

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

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

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

573 **6.3.3 Primordial resource pool**

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

577 **6.3.4 Concrete resource pool**

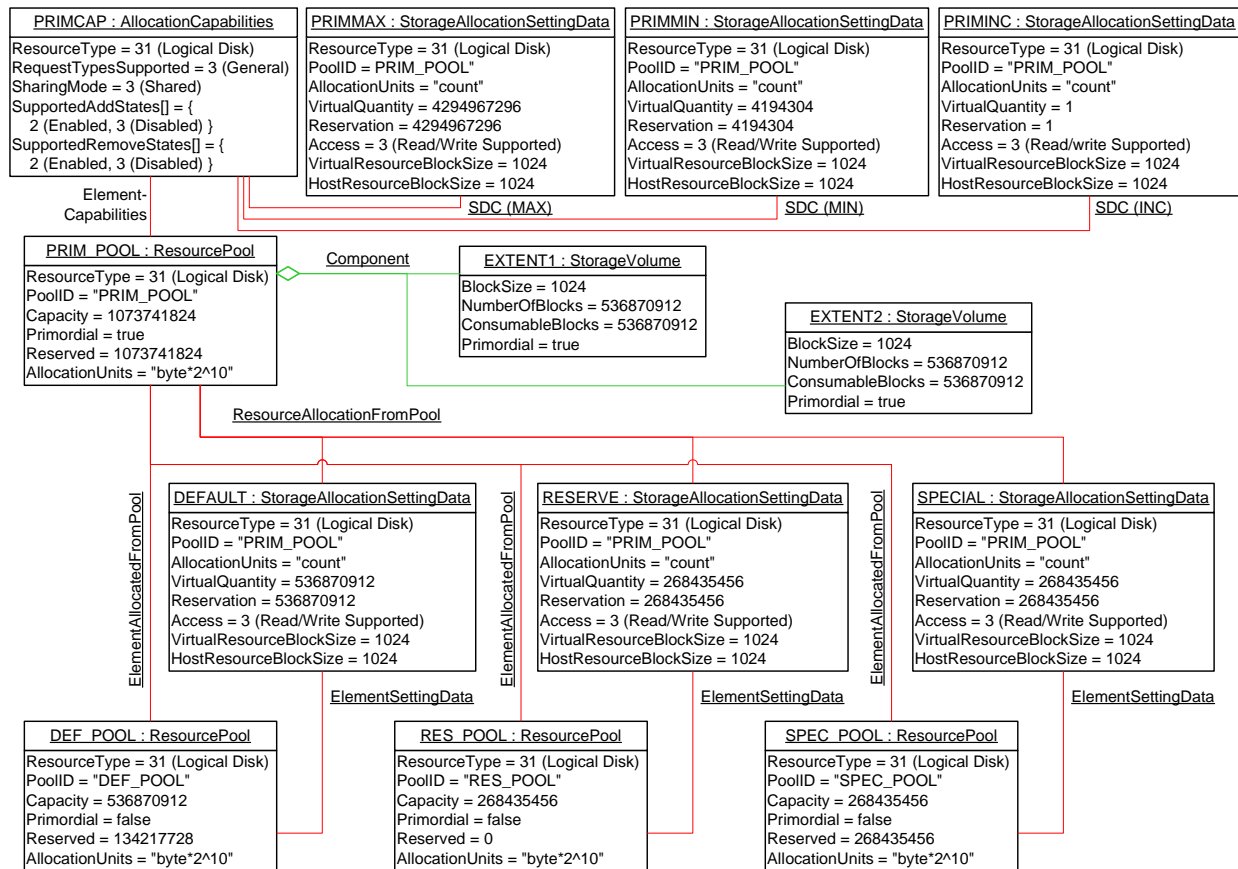
578 A concrete resource pool subdivides the capacity of its parent resource pool. The amount of capacity
579 allocated to a concrete resource pool is less than or at most equal to the capacity of the parent pool.

580 **6.3.5 Hierarchies of resource pools**

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

584 Figure 2 shows an example of the CIM representation of a resource pool hierarchy where a set of host
585 storage extents is aggregated into a primordial storage resource pool PRIM_POOL. The allocation
586 capabilities of PRIM_POOL are represented by means of [DMTF DSP1043:1.0](#) (*Allocation Capabilities
587 Profile*). PRIM_POOL supports allocations starting from 4 GB up to 4 TB; in that range the increment is
588 1 KB.

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



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

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

- 597
- 598
- 599
- 600
- 601
- 602
- 603
- 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.

604 6.3.6 Resource pool management

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

608 6.4 Resource allocation

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

612 **6.4.1 General**

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

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

- 618 1) The allocation of a storage extent without explicit representation of a disk drive
- 619 2) The allocation of a storage extent with explicit representation of a disk drive

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

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

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

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

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

634 **6.4.2 Storage resource allocations backed by files**

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

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

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

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

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

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

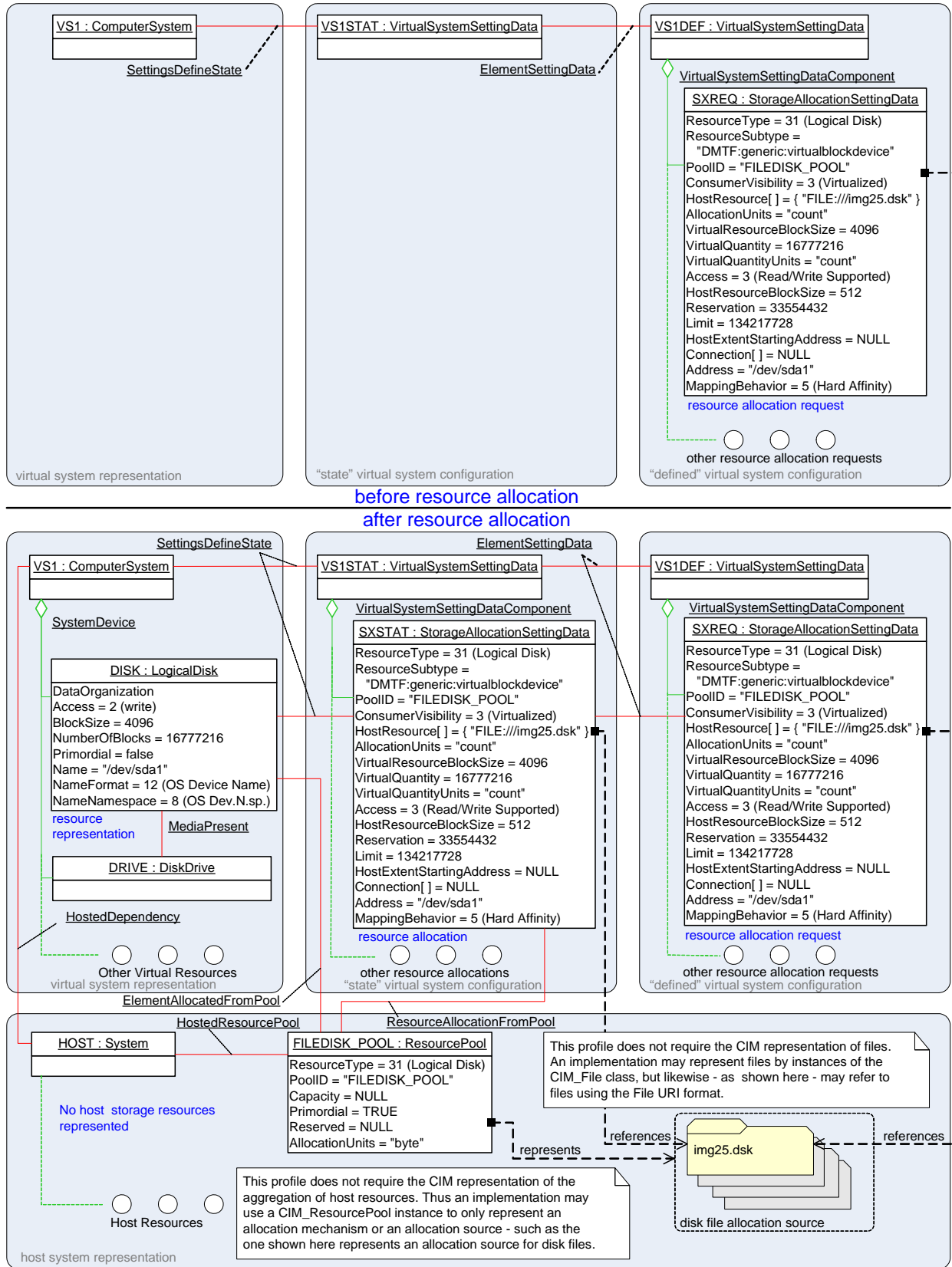
652 The concatenation of the parts yields `"/var/vmfiles/vm1/disks/imagedisk25.dsk"`. Note that such rules are
653 highly implementation dependent. The model defined in this profile facilitates the representation of rule-
654 based assignment through the means of resource pools, but it does not specify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

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

667 The example shown in Figure 3 also shows a representation of thin provisioning of a file-based logical
668 disk. This is indicated by the value of the `Limit` property being defined and higher than the value of the
669 `Reservation` property. In this example the value of the `Reservation` property requests an initial file size of
670 33554432 blocks (16 GB) up to a maximum file size of 134217728 blocks (64 GB) as expressed by the
671 value of the `Limit` property; in both cases a block size of 512 applies as expressed by the value of the
672 `HostResourceBlockSize` property. Opposed to that the disk size as seen by the virtual system (the
673 consumer) remains constant at 64 GB; this is expressed by the value of the `VirtualQuantity` property,
674 16777216 blocks with a block size of 4096 as expressed by the value of the `VirtualResourceBlockSize`
675 property. As the consuming virtual system starts writing data onto the virtual disk, for each logical 4-KB
676 block of virtual disk a respective set of eight 512-KB blocks is allocated within the file. As soon as the
677 amount of data to back the logical disk exceeds the initially requested file size (as expressed by the value
678 of the `Reservation` property) the file starts growing beyond the initially assigned file size up to the size
679 expressed by the value of the `Limit` property, such that finally when all blocks were at least once written
680 by the virtual system the amount of storage provided equals the amount of storage consumed.

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

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



692

693

Figure 3 – Instance diagram: Concept of storage resource allocation

694 **6.4.3 Resource allocation request**

695 The resource requirements of a virtual system are represented by the "defined" virtual system
696 configuration (see [DMTF DSP1057:1.0 \(Virtual System Profile\)](#)). In a "defined" virtual system
697 configuration disk drive resource allocation requests are represented by
698 CIM_ResourceAllocationSettingData instances, and storage resource allocation requests are represented
699 by CIM_StorageAllocationSettingData instances.

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

702 **6.4.4 Resource allocation**

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

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

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

719 **6.4.5 Virtual disk**

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

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

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

726 **6.4.6 Virtual disk drive**

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

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

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

736 **6.4.7 Storage virtualization**

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

740 **6.4.8 Dedicated host storage**

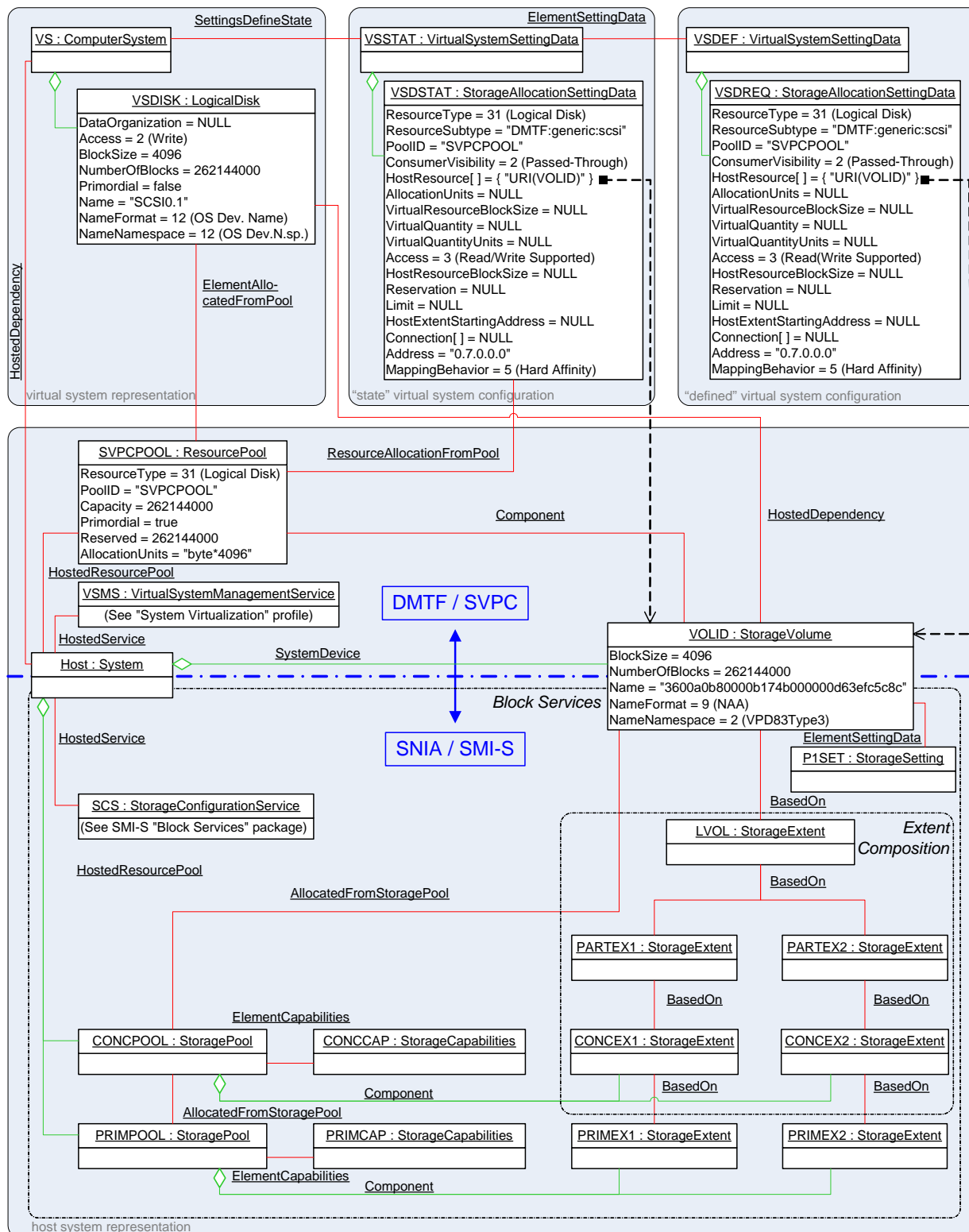
741 A dedicated storage extent is a storage extent owned or accessible by the host system that is exclusively
742 reserved for support of a particular virtual disk of a particular virtual system.

743 **6.4.9 Virtual storage extended configurability**

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

751 **6.4.10 Management of host storage resources through SMI-S profiles**

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



756

757

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

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

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

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

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

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

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

790 **7 Implementation**

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

793 **7.1 Common requirements**

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

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

799 Implementations of this profile shall expose an instance of the CIM_RegisteredProfile class as adapted in
800 10.11 in the Interop namespace. That instance shall be associated with the CIM_RegisteredProfile
801 instance representing the implementation of the scoping profile through an instance of the
802 CIM_ReferencedProfile association as adapted in 10.10. Additional instance requirements specified in
803 [DMTF DSP1033:1.0 \(Profile Registration Profile\)](#) may apply.

804 **7.2 Resource types**

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

806 7.2.1 General

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

809 7.2.2 Logical disks, storage volumes and storage extents

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

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

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

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

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

829 7.2.3 Disk drives

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

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

834 7.3 Host resources

835 This subclause specifies requirements for the representation of host resources.

836 7.3.1 Host storage volume

837 The representation of host storage volumes is conditional.

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

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

845 7.3.2 Host disk drives

846 The representation of host disk drives is conditional.

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

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

854 7.4 Resource pools

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

857 7.4.1 General

858 Implementations of this profile for one of the resource types 31 (Logical Disk), 32 (Storage Volume) or
859 19 (Storage Extent) may choose to implement the CIM_StoragePool class (which is a subclass of the
860 CIM_ResourcePool class) in place of the CIM_ResourcePool class. The provisions in this subclause
861 apply likewise to implementations of the CIM_ResourcePool class itself, or to implementations of the
862 CIM_StoragePool class if that is implemented instead of the CIM_ResourcePool class. This profile does
863 not adapt properties defined by the CIM_StoragePool class.

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

868 7.4.2 ResourceType property

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

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

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

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

880 7.4.3 ResourceSubType property

881 The implementation of the ResourceSubType property is optional.

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

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

887 EXPERIMENTAL

888 **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

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

891 EXPERIMENTAL

892 The implementation should apply the mechanisms defined in [DMTF DSP1043:1.0 \(Allocation Capabilities
893 Profile\)](#) to expose the resource subtypes that are supported by the implementation.

894 **7.4.4 Primordial property**

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

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

899 **7.4.5 PoolID property**

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

902 **7.4.6 Reserved property**

903 The implementation of the Reserved property is optional.

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

- 906 • If the value of the ResourceType property is any of 31 (Logical Disk), 32 (Storage Volume) or
907 19 (Storage Extent), the value of the Reserved property shall reflect the amount of storage that
908 is allocated from the resource pool, in units as specified by the value of the AllocationUnits
909 property (see 7.4.8).

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

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

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

915 **7.4.7 Capacity property**

916 The implementation of the Capacity property is conditional.

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

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

- 920 • If the value of the ResourceType property is any of 31 (Logical Disk), 32 (Storage Volume) or
921 19 (Storage Extent), the value of the Capacity property shall reflect the maximum amount of
922 storage that can be allocated from the resource pool, in units as specified by the value of the
923 AllocationUnits property.
- 924 • If the value of the ResourceType property is 17 (Disk Drive), the value of the Capacity property
925 shall reflect the maximum number of disk drives that can be allocated from the resource pool.

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

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

929 **7.4.8 AllocationUnits property**

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

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

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

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

942 **7.4.9 MaxConsumableResource property**

943 The implementation of the MaxConsumableResource property is optional.

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

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

951 **7.4.10 CurrentlyConsumedResource property**

952 The implementation of the CurrentlyConsumedResource property is optional.

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

956 **7.4.11 ConsumedResourceUnit property**

957 The implementation of the ConsumedResourceUnit property is conditional.

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

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

963 **7.4.12 Instance requirements**

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

965 **7.5 Resource aggregation feature**

966 The implementation of the resource aggregation feature is conditional.

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

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

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

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

977 **7.6 Resource pool hierarchies feature**

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

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

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

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

988 NOTE: The [SNIA SMIS:1.3, Part 3 Block Devices, Block Services](#) package requires the
989 implementation of the CIM_ConcreteComponent association for the representation of the same
990 relationship if the [SNIA SMIS:1.3, Part 3 Block Devices, Extent Composition](#) subprofile is

991 implemented; in this case implementations of the [SNIA SMIS: 1.3, Part 3 Block Devices, Block](#)
 992 [Services](#) package need to implement both associations.

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

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

1000 NOTE: If for example the Associators() intrinsic operation is used to resolve the association, the Role parameter
 1001 or the ResultRole parameters may be used to distinguish the parent-to-child relationship from the child-to-
 1002 parent relationship.

1003 7.7 Resource pool management feature

1004 The implementation of the resource pool management feature is optional.

1005 If implemented, the specifications of [DMTF DSP1041:1.1 \(Resource Allocation Profile\)](#) apply; this profile
 1006 does not specify specializations or extensions of resource pool management beyond those defined by
 1007 [DMTF DSP1041:1.1](#).

1008 7.8 Resource allocation

1009 This subclause details requirements for the representation of resource allocation information through
 1010 CIM_ResourceAllocationSettingData (RASD) instances or CIM_StorageAllocationSettingData (SASD)
 1011 instances.

1012 7.8.1 General

1013 Implementations of this profile shall implement the virtual resource allocation pattern as defined in [DMTF](#)
 1014 [DSP1041:1.1](#), subclause 7.2 (Virtual resource allocation).

1015 NOTE: [DMTF DSP1041:1.1](#) specifies two alternatives for modeling resource allocation: *simple resource*
 1016 *allocation* and *virtual resource allocation*.

1017 This profile adapts the CIM_StorageAllocationSettingData (SASD) class for storage resource allocation
 1018 and the CIM_ResourceAllocationSettingData class for disk drive resource allocation.

1019 7.8.2 Flavors of allocation data

1020 Various flavors of allocation data describes are defined:

- 1021 • Resource allocation requests; for details see 6.4.3.
- 1022 • Resource allocations; for details see 6.4.4.
- 1023 • Settings that define the capabilities or mutability of managed resources. [DMTF DSP1043:1.0](#)
 1024 specifies a capabilities model that conveys information about the capabilities and the mutability
 1025 of managed resources in terms of RASD instances.
- 1026 • Parameters in operations that define or modify any of the representations listed above. [DMTF](#)
 1027 [DSP1042:1.0](#) that specifies methods for the definition and modification of virtual resources.
 1028 These methods use RASD instances for the parameterization of resource-allocation-specific
 1029 properties.

1030 Table 4 lists acronyms that are used in subclauses of 7.8 in order to designate RASD or SASD instances
 1031 that represent various flavors of allocation data.

1032 **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

1033 Subclauses of 7.8 detail implementation requirements for property values in RASD instances. In some
 1034 cases requirements only apply to a subset of the flavors listed in Table 4; this is marked in the text
 1035 through the use of respective acronyms.

1036 **7.8.3 CIM_ResourceAllocationSettingData properties**

1037 This subclause defines rules for values of properties in instances of the
 1038 CIM_ResourceAllocationSettingData (RASD) class representing disk drive allocation information.

1039 **7.8.3.1 ResourceType property**

1040 The value of the ResourceType property in RASD instances representing disk drive allocation information
 1041 shall be set to 17 (Disk Drive) for disk drive allocation data.

1042 Other values shall not be used.

1043 **7.8.3.2 ResourceSubType property**

1044 The implementation of the ResourceSubType property is optional.

1045 If the ResourceSubType property is implemented, the provisions defined for the ResourceSubType
 1046 property of the CIM_ResourcePool class; see 7.4.2.

1047 **7.8.3.3 PoolID property**

1048 The value of the PoolID property shall identify the resource pool. The special value NULL shall indicate
 1049 the use of the host system's default resource pool for the selected resource type.

1050 **7.8.3.4 ConsumerVisibility property**

1051 The implementation of the ConsumerVisibility property is optional.

1052 If the ConsumerVisibility property is implemented, the provisions in this subclause apply.

1053 The value of the ConsumerVisibility property shall denote either if a host resources is directly passed
1054 through to the virtual system as a virtual resource, or if the resource is virtualized. Values shall be set as
1055 follows:

- 1056 • A value of 2 (Passed-Through) shall denote that the host resource is passed-through.
- 1057 • A value of 3 (Virtualized) shall denote that the virtual resource is virtualized.
- 1058 • Only in instances of { Q_RASD | D_RASD | M_RASD }, the special value NULL shall be used if
1059 the represented resource allocation request does not predefine which kind of consumer visibility
1060 (passed-through or virtualized) is requested.
- 1061 • Other values shall not be used.

1062 **7.8.3.5 HostResource[] array property**

1063 The implementation of the HostResource[] array property is conditional.

1064 Condition: The HostResource[] array property shall be implemented if any of the following conditions is
1065 true: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property, or any
1066 of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the MappingBehavior
1067 property.

1068 If the HostResource[] array property is implemented, the provisions in this subclause apply.

1069 In the cases of { Q_RASD | C_RASD | D_RASD | M_RASD } the value of the HostResource[] array
1070 property shall refer to the representation of one or more host resources that are configured to contribute
1071 to the disk drive resource allocation. In the case of R_RASD the value of the HostResource[] array
1072 property shall refer to a representation of the host resource that provides the disk drive resource
1073 allocation.

1074 Elements of the value of the HostResource[] array property shall refer to instances of CIM classes, using
1075 the WBEM URI format as specified by [DMTF DSP0207](#). Referenced instances shall be of the
1076 CIM_CDROMDrive class, the CIM_DiskDrive class, the CIM_DisketteDrive class, the CIM_DVDDrive
1077 class or the CIMWORMDrive class.

1078 **7.8.3.6 AllocationUnits property**

1079 The value of the AllocationUnits property shall be "count".

1080 NOTE: The units defined by value of the AllocationUnits property applies to the values of the Reserved and the
1081 Limit property; it does not apply to the value of the VirtualQuantity property.

1082 **7.8.3.7 VirtualQuantity property**

1083 The value of the VirtualQuantity property shall denote the number of virtual disk drives available to a
1084 virtual system through this resource allocation.

1085 EXPERIMENTAL

1086 **7.8.3.8 VirtualQuantityUnits property**

1087 The value of the VirtualQuantityUnits property shall be "count".

1088 EXPERIMENTAL

1089 **7.8.3.9 Reservation property**

1090 The implementation of the Reservation property is optional.

1091 If the Reservation property is implemented, the provisions in this subclause apply.

1092 The value of the Reservation property shall denote the number of disk drives reserved through this
1093 resource allocation to a virtual system.

1094 **7.8.3.10 Limit property**

1095 The implementation of the Limit property is optional.

1096 If the Limit property is implemented, the following rules apply:

1097 The value of the Limit property shall denote the maximum number of disk drives available through this
1098 resource allocation to a virtual system.

1099 **7.8.3.11 Weight property**

1100 The implementation of the Weight property is optional.

1101 If the Weight property is implemented, its value shall denote the relative priority of a resource allocation in
1102 relation to other resource allocations.

1103 **7.8.3.12 Parent property**

1104 The implementation of the Parent property is optional.

1105 If the Parent property is implemented, the provisions in this subclause apply.

1106 The value of the Parent property shall identify the parent entity of the resource allocation or resource
1107 allocation request. The value of the Parent property shall be formatted with the WBEM URI format as
1108 specified by [DMTF DSP0207](#).

1109 If an implementation implements the concept of disk snapshots where data stored on a delta disk only
1110 contains information on top of that stored on a base disk, then the implementation should use the value of
1111 the Parent property in the RASD instance representing the storage resource allocation of the delta disk to
1112 refer to the RASD instance representing the storage resource allocation of the base disk.

1113 **7.8.3.13 Connection[] array property**

1114 The implementation of the Connection[] array property is optional.

1115 If the Connection[] array property is implemented, the provisions in this subclause apply.

1116 The value of the connection property may identify elements of the storage infrastructure such as initiator
1117 ports and/or target ports. The WBEM URI format (see [DMTF DSP0207](#)) may be used to refer to a
1118 respective CIM instance.

1119 7.8.3.14 Address property

1120 The implementation of the Address property is optional.

1121 If the Address property is implemented, the provisions in this subclause apply.

1122 The value of the Address property shall expose the address of the allocated resource as seen by the
1123 software running in the virtual system (usually a guest operating system).

1124 7.8.3.15 MappingBehavior property

1125 The implementation of the MappingBehavior property is optional.

1126 If the MappingBehavior property is implemented, its value shall denote how host resources referenced by
1127 elements in the value of HostResource[] array property relate to the resource allocation.

1128 In R_RASD instances the following rules apply to the value of the MappingBehavior property:

- 1129 • A value of 2 (Dedicated) shall indicate that the represented resource allocation is provided by
1130 host resources that are exclusively dedicated to the virtual system. The host resources shall be
1131 identified by the value of the HostResource[] array property.
- 1132 • A value of 3 (Soft Affinity) or 4 (Hard Affinity) shall indicate that the represented resource
1133 allocation is provided by host resources. The host resources shall be identified by the value of
1134 the HostResource[] array property.
- 1135 • Other values shall not be used.

1136 In Q_RASD instances the following rules apply to the value of the MappingBehavior property:

- 1137 • The special value NULL shall indicate that the resource allocation request does not require
1138 specific host resources.
- 1139 • A value of 2 (Dedicated) shall indicate that the resource allocation request shall be provided by
1140 exclusively dedicated host resources as specified through the value of the HostResource[]
1141 array property.
- 1142 • A value of 3 (Soft Affinity) shall indicate that the resource allocation request shall preferably be
1143 provided by host resources as specified through the value of the HostResource[] array
1144 property, but that other host resources may be used if the requested host resources are not
1145 available.
- 1146 • A value of 4 (Hard Affinity) shall indicate that the resource allocation request shall be provided
1147 by host resources as specified through the value of the HostResource[] array property and that
1148 other resources shall not be used if the requested host resources are not available.
- 1149 • Other values shall not be used.

1150 7.8.4 CIM_StorageAllocationSettingData properties

1151 This subclause defines rules for values of properties in instances of the
1152 CIM_StorageAllocationSettingData (SASD) class.

1153 NOTE: If the rules for a particular property are the same as those defined for the respective property of the
1154 CIM_ResourceAllocationSettingData (RASD) class, the respective subclause of 7.8.3 is referenced.

1155 7.8.4.1 ResourceType property

1156 The value of the ResourceType property shall be set as follows:

- 1157 • 31 (Logical Disk) for logical disk allocation data
- 1158 • 32 (Storage Volume) for storage volume allocation data

- 19 (Storage Extent) for storage extent allocation data

1160 NOTE: See 7.2.2 for a definition of logical disk, storage volume and storage extent.

1161 **7.8.4.2 ResourceSubtype property**

1162 See 7.8.3.2.

1163 **7.8.4.3 PoolID property**

1164 See 7.8.3.3.

1165 **7.8.4.4 ConsumerVisibility property**

1166 See 7.8.3.4.

1167 **7.8.4.5 HostResource array property**

1168 The implementation of the HostResource[] array property is conditional.

1169 Condition: The HostResource[] array property shall be implemented if any of the following conditions is
1170 true: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property, or any
1171 of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the MappingBehavior
1172 property.

1173 If the HostResource[] array property is implemented, the provisions in this subclause apply.

1174 In the cases of { Q_SASD | C_SASD | D_SASD | M_SASD } the value of the HostResource[] array
1175 property shall refer to (the representation of) one or more host resources that are configured to contribute
1176 to the resource allocation. In the case of R_SASD the value of the HostResource[] array property shall
1177 refer to (the representation of) the host resource that provides the storage resource allocation.

1178 Values of elements of the HostResource[] array property may directly refer to files, using the URI format
1179 as specified by [IETF RFC1738](#) and file URL scheme as specified in [IETF RFC3986](#).

1180 If the file URI is not applied, elements of the value of the HostResource[] array property shall refer to
1181 instances of CIM classes, using the Wbem URI format as specified by [DMTF DSP0207](#). Referenced
1182 instances shall be of the CIM_StorageExtent class or the CIM_LogicalFile class.

1183 **7.8.4.6 AllocationUnits property**

1184 The implementation of the AllocationUnits property is conditional.

1185 Condition: The Reservation property (see 7.8.4.9) or the Limit property (see 7.8.4.10), or both are
1186 implemented.

1187 The AllocationUnits property shall convey the unit applicable to the values of the Reservation and the
1188 Limit property.

1189 If the AllocationUnits property is implemented, the provisions in this subclause apply.

1190 If the value of the BlockSize property is 1, the value of the AllocationUnits property shall be "byte",
1191 indicating that the values of the Reservation and of the Limit property are specified in bytes. If the
1192 value of the BlockSize property is greater than 1, the value of the AllocationUnits property shall be
1193 "count", indicating that the values of the Reservation and of the Limit property are specified in blocks, with
1194 the blocksize conveyed through the value of the BlockSize property.

1195 All flavors of SASD instances as defined in 7.8.2 that relate to the same virtual resource shall apply the
1196 same value for the AllocationUnits property.

1197 NOTE: The definitions in this subclause include SASD instances that describe mutability. In these instances the
 1198 mutability is expressed by values of numerical properties such as the Reservation or the Limit property in
 1199 units as established by the value of the AllocationUnit property. If the mutability SASD instance represents
 1200 an increment, this would reflect a granularity for modifications of the numeric property values that is equal
 1201 to or a multiple of the allocation unit.

1202 **7.8.4.7 VirtualQuantity property**

1203 In the cases of { R_SASD | C_SASD | D_SASD | M_SASD } the value of the VirtualQuantity property
 1204 shall denote the amount of storage that is available to a virtual system through this resource allocation. In
 1205 the case of Q_SASD the value of the VirtualQuantity property shall denote the amount of storage that is
 1206 requested for the virtual system unless the value of the HostResource[] array property contains exactly
 1207 one element that refers to a specific host storage resource that implicitly determines the virtual disk size.
 1208 If a value is provided, it shall be expressed in units as expressed by the value of the VirtualQuantityUnit
 1209 property; see 7.8.4.8.

1210 EXPERIMENTAL

1211 **7.8.4.8 VirtualQuantityUnits property**

1212 The VirtualQuantityUnits property shall convey the unit applicable to the value of the VirtualQuantity
 1213 property.

1214 If the value of the VirtualResourceBlockSize property is 1, the value of the VirtualQuantityUnits property
 1215 shall be "byte", indicating that the value of the VirtualQuantity property is specified in bytes. If the
 1216 value of the VirtualBlockSize property is greater than 1, the value of the VirtualQuantityUnits property
 1217 shall be "count", indicating that the value of the VirtualQuantity property is specified in blocks, with the
 1218 blocksize conveyed through the value of the VirtualResourceBlockSize property.

1219 EXPERIMENTAL

1220 **7.8.4.9 Reservation property**

1221 The implementation of the Reservation property is optional.

1222 If the Reservation property is implemented, the provisions in this subclause apply.

1223 The value of the Reservation property shall denote the amount of storage reserved through this resource
 1224 allocation to a virtual system in units as expressed by the value of the AllocationUnits property;
 1225 see 7.8.4.6.

1226 **7.8.4.10 Limit property**

1227 The implementation of the Limit property is optional.

1228 If the Limit property is implemented, the provisions in this subclause apply.

1229 The value of the Limit property shall denote the maximum amount of storage available through the
 1230 represented resource allocation to a virtual system in units as expressed by the value of the
 1231 AllocationUnits property; see 7.8.4.6.

1232 **7.8.4.11 Weight property**

1233 See 7.8.3.11.

1234 **7.8.4.12 Parent property**

1235 See 7.8.3.12.

1236 7.8.4.13 Connection[] array property

1237 See 7.8.3.13.

1238 7.8.4.14 Address property

1239 See 7.8.3.14.

1240 7.8.4.15 MappingBehavior property

1241 See 7.8.3.15.

1242 7.8.4.16 VirtualResourceBlockSize

1243 The value of the VirtualResourceBlockSize property shall denote the block size as seen by the consumer
1244 of a virtual storage that is based on the described resource allocation. A value of 1 shall designate a
1245 variable block size.

1246 7.8.4.17 Access

1247 The value of the Access property shall denote the access mode.

1248 7.8.4.18 HostResourceBlockSize

1249 The value of the HostResourceBlockSize property shall denote the block size as seen by the consumer of
1250 a virtual storage that is based on the described resource allocation. A value of 1 shall designate a
1251 variable block size.

1252 7.8.4.19 HostExtentStartingAddress

1253 The implementation of the HostExtentStartingAddress property is optional.

1254 If the HostExtentStartingAddress property is implemented, the provisions in this subclause apply.

1255 The value of the HostExtentStartingAddress property shall denote the offset within the host storage extent
1256 referenced by the value of the HostExtentName property. The offset marks the starting point of a
1257 subspace within the referenced host storage extent. The size of the subspace is exposed by the value of
1258 the Reserved property.

1259 7.8.4.20 HostExtentName

1260 The implementation of the HostExtentName property is optional.

1261 If the HostExtentName property is implemented, the provisions in this subclause apply.

1262 The value of the HostExtentName shall identify a host storage extent that serves as the base for the
1263 described virtual storage allocation.

1264 7.8.4.21 HostExtentNameFormat

1265 The implementation of the HostExtentNameFormat property is conditional.

1266 Condition: The HostExtentName property (see 7.8.4.20) is implemented.

1267 If the HostExtentNameFormat property is implemented, the provisions in this subclause apply.

1268 The value of the HostExtentNameFormat shall designate the format used for the value of the
1269 HostExtentName property.

1270 7.8.4.22 OtherHostExtentNameFormat

1271 The implementation of the HostExtentNameFormat property is conditional.

1272 Condition: The HostExtentNameFormat property (see 7.8.4.21) is implemented, and the value 1 (Other) is
1273 supported.

1274 If the OtherHostExtentNameFormat property is implemented, the provisions in this subclause apply.

1275 The value of the HostExtentNameFormat shall designate the format used for the value of the
1276 HostExtentName property, using a string representation. The value should be structured as follows:
1277 <Organization>:<FormatSpecifier>. <Organization> shall uniquely identify the organization that defined
1278 the format. <FormatSpecifier> shall uniquely identify the format within the set of formats defined by the
1279 organization.

1280 7.8.4.23 HostExtentNameNamespace

1281 The implementation of the HostExtentNameNamespace property is conditional.

1282 Condition: The HostExtentName property (see 7.8.4.20) is implemented.

1283 If the HostExtentNameNamespace property is implemented, the provisions in this subclause apply.

1284 The value of the HostExtentNameNamespace shall designate the namespace that applies to the value of
1285 the HostExtentName property.

1286 7.8.4.24 OtherHostExtentNameNamespace

1287 The implementation of the OtherHostExtentNameNamespace property is conditional.

1288 Condition: The HostExtentNameNamespace property (see 7.8.4.23) is implemented, and the value
1289 1 (Other) is supported.

1290 If the OtherHostExtentNameNamespace property is implemented, the provisions in this subclause apply.

1291 The value of the HostExtentNameNamespace shall designate the namespace used for the value of the
1292 HostExtentName property, using a string representation. The value should be structured as follows:
1293 <Organization>:<NamespaceSpecifier>. <Organization> shall uniquely identify the organization that
1294 defined the format. <NamespaceSpecifier> shall uniquely identify the namespace within the set of
1295 namespaces defined by the organization.

1296 7.8.5 Instance requirements

1297 This subclause details resource allocation related instance requirements.

1298 7.8.5.1 Representation of resource allocation requests

1299 If this profile is implemented for the allocation of storage extents (see 7.2), each storage resource
1300 allocation request shall be represented by a Q_SASD instance; the provisions of 10.15 apply.

1301 If this profile is implemented for the allocation of disk drives (see 7.2), each disk drive resource allocation
1302 request shall be represented by a Q_RASD instance; the provisions of 10.12 apply.

1303 7.8.5.2 Representation of resource allocations

1304 If this profile is implemented for the allocation of storage extents (see 7.2), each storage resource
1305 allocation shall be represented by a R_SASD instance; the provisions of 10.15 apply.

1306 If this profile is implemented for the allocation of disk drives (see 7.2), each disk drive resource allocation
1307 shall be represented by a R_RASD instance; the provisions of 10.12 apply.

1308 The R_SASD (or R_RASD) instance shall be associated to the Q_SASD (or Q_RASD) instance
1309 representing the corresponding resource allocation request (see 7.8.5.1) through an instance of the
1310 CIM_ElementSettingData association; the provisions of 10.6 apply.

1311 The R_SASD (or R_RASD) instance shall be associated to the CIM_ResourcePool instance providing
1312 resources for the allocation (see 7.4) through an instance of the CIM_ResourceAllocationFromPool
1313 association; the provisions of 10.6 apply.

1314 Implementations may represent a resource allocation request and the corresponding resource allocation
1315 by one SASD (or RASD) instance; in this case the association requirements of this subclause apply
1316 correspondingly. Note that association instances that refer to the R_SASD instance are only existent
1317 while the resource is allocated.

1318 **7.8.5.3 Representation of resource allocation capabilities**

1319 The allocation capabilities of a system or a resource pool shall be represented by an
1320 CIM_AllocationCapabilities instance that is associated to the CIM_System instance representing the
1321 system or to the CIM_ResourcePool instance representing the resource pool through an instance of the
1322 CIM_ElementCapabilities association; see [DMTF DSP1043:1.0 \(Allocation Capabilities Profile\)](#).

1323 The settings that define the allocation capabilities of a storage resource pool shall be represented by
1324 C_SASD instances; the provisions of 10.15 apply.

1325 The settings that define the allocation capabilities of a disk drive resource pool shall be represented by
1326 C_RASD instances; the provisions of 10.12 apply.

1327 **7.8.5.4 Representation of resource allocation mutability**

1328 The mutability of a resource allocation or resource allocation request shall be represented by an
1329 CIM_AllocationCapabilities instance that is associated to the RASD instance representing the resource
1330 allocation of resource allocation request through an instance of the CIM_ElementCapabilities association;
1331 see [DMTF DSP1043:1.0 \(Allocation Capabilities Profile\)](#).

1332 The settings that define the allocation capabilities of a storage resource pool shall be represented by
1333 C_SASD instances; the provisions of 10.15 apply.

1334 The settings that define the allocation capabilities of a disk drive resource pool shall be represented by
1335 C_RASD instances; the provisions of 10.12 apply.

1336 **7.9 Virtual resources**

1337 This subclause specifies rules for the representation of virtual resources. Virtual resources are the result
1338 of a resource allocations. Virtual resources are scoped by virtual systems or by virtual storage arrays.
1339 Virtual storage arrays are a special kind of virtual system that serve the purpose of providing storage to
1340 other virtual systems.

1341 **7.9.1 Virtual resource instance requirements**

1342 An allocated virtual resource shall be represented by an instance of a subclass of the CIM_LogicalDevice
1343 class, as follows:

- 1344 • Virtual disks

1345 The representation of virtual disks is governed by the type of virtual disk as defined in 7.2.2.

- 1346 – An allocated virtual disk shall be represented by a CIM_StorageExtent instance if the value
1347 of the ResourceType property in the CIM_StorageAllocationSettingData instance
1348 representing the virtual disk allocation is 19 (Storage Extent). However, if the allocated

- 1349 virtual disk conforms to the stricter definitions of logical disk or storage volume (see 7.2.2),
1350 it may be represented by a CIM_LogicalDisk or a CIM_StorageVolume, respectively.
- 1351 – A allocated virtual disk shall be represented by a CIM_StorageVolume instance if the value
1352 of the ResourceType property in the CIM_StorageAllocationSettingData instance
1353 representing the virtual disk allocation is 32 (Storage Volume)
 - 1354 – A allocated virtual disk shall be represented by a CIM_LogicalDisk instance if the value of
1355 the ResourceType property in the CIM_StorageAllocationSettingData instance
1356 representing the virtual disk allocation is 31 (Logical Disk).
- 1357 • Virtual disk drives
- 1358 A virtual disk drive shall be represented by an instance of the CIM_DiskDrive class
- 1359 • Virtual ports
- 1360 If implemented, virtual initiator ports or virtual target ports shall be represented by instances of
1361 the CIM_LogicalPort class
- 1362 Each instance of a subclass of the CIM_LogicalDevice class representing a virtual resource as defined in
1363 this subclause shall be associated as follows:
- 1364 • to the CIM_ComputerSystem instance that represents the virtual system through an instance of
1365 the CIM_SystemDevice association
 - 1366 • to the SASD or RASD instance that represents the resource allocation through an instance of
1367 the CIM_SettingsDefineState association
- 1368 NOTE: If the resource allocation of a logical disk is a composed resource allocation, only the top-most
1369 resource allocation yields a logical device. Consequently there may be SASD instances within a
1370 "state" virtual system configuration that do not have a companion logical device.
- 1371 • to the CIM_ResourcePool instance that represents the resource pool providing the resource
1372 allocation through an instance of the CIM_ElementAllocatedFromPool association.

1373 7.9.2 CIM_StorageExtent properties

1374 This subclause defines constraints for property values in CIM_StorageExtent instances representing
1375 virtual disks.

1376 7.9.2.1 BlockSize

1377 The value of the BlockSize property shall be identical to the value of the VirtualResourceBlockSize
1378 property in the SASD instance representing the related storage resource allocation (see 7.8.4.16).

1379 7.9.2.2 NumberOfBlocks

1380 The value of the NumberOfBlocks property shall be identical to the value of the VirtualQuantity property in
1381 the SASD instance representing the related storage resource allocation (see 7.8.4.7).

1382 7.9.2.3 Name

1383 The value of the Name property shall expose the name of the storage extent as seen by the virtual
1384 system.

1385 7.9.2.4 NameFormat

1386 The value of the NameFormat property shall be 12 (OS Device Name).

1387 7.9.2.5 NameNamespace

1388 The value of the NameFormat property shall be 8 (OS Device Namespace).

1389 8 Methods

1390 This section details the requirements for supporting intrinsic operations and extrinsic methods for the CIM
1391 elements defined by this profile.

1392 8.1 Profile conventions for operations

1393 The implementation requirements on intrinsic operations for each profile class (including associations) are
1394 specified in class specific subclauses of this clause.

1395 The default list of intrinsic operations for all classes is:

- 1396 • GetInstance()
- 1397 • EnumerateInstances()
- 1398 • EnumerateInstanceNames()

1399 For classes that are referenced by an association, the default list also includes

- 1400 • Associators()
- 1401 • AssociatorNames()
- 1402 • References()
- 1403 • ReferenceNames()

1404 Implementation requirements on operations defined in the default list are provided in the class-specific
1405 subclauses of this clause.

1406 The implementation requirements for intrinsic and extrinsic methods of classes listed in clause 10, but not
1407 addressed by a separate subclause of this clause are specified by the "Methods" clauses of respective
1408 base profiles, namely [DMTF DSP1041:1.1](#) (*Resource Allocation Profile*) and [DMTF DSP1043:1.0](#)
1409 (*Allocation Capabilities Profile*). These profiles are specialized by this profile, and in these cases this
1410 profile does not add method specifications beyond those defined in its base profiles.

1411 8.2 CIM_DiskDrive for host disk drives

1412 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1413 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1414 apply.

1415 8.3 CIM_DiskDrive for virtual disk drives

1416 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1417 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1418 apply.

1419 8.4 CIM_LogicalDisk for virtual disk drives

1420 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1421 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1422 apply.

1423 **8.5 CIM_ReferencedProfile**

1424 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1425 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1426 apply.

1427 **8.6 CIM_RegisteredProfile**

1428 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1429 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1430 apply.

1431 **8.7 CIM_StorageAllocationSettingData for storage extent allocation information**

1432 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#):1.3.
1433 In addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1434 apply.

1435 **8.8 CIM_StorageExtent for virtual disk**

1436 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1437 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1438 apply.

1439 **8.9 CIM_SystemDevice for host storage volumes**

1440 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1441 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1442 apply.

1443 **8.10 CIM_SystemDevice for virtual resources**

1444 All intrinsic operations in the default list in 8.1 shall be implemented as specified by [DMTF DSP0200](#). In
1445 addition, the requirements of the CIM schema and other prerequisite specifications (including profiles)
1446 apply.

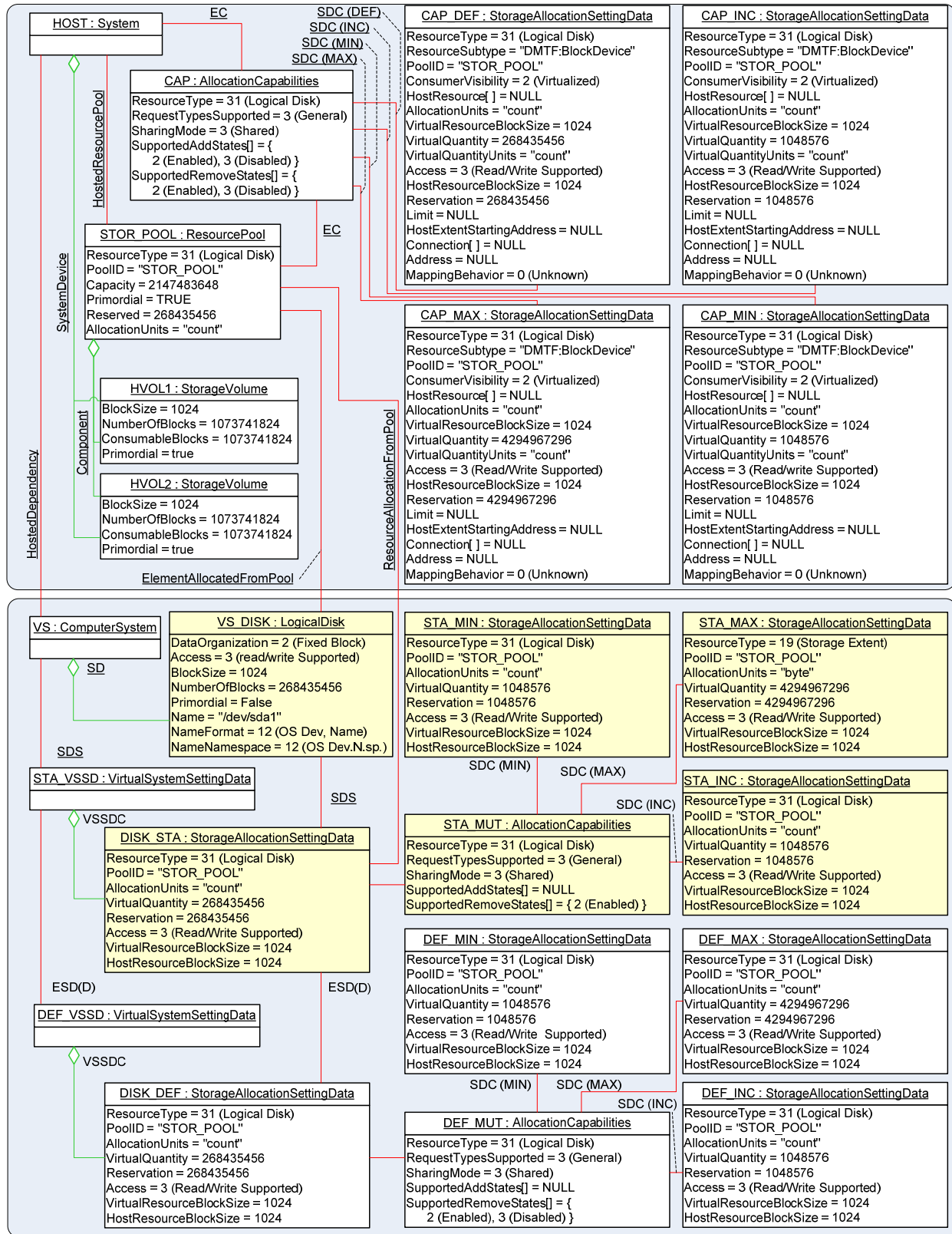
1447 **9 Use cases**

1448 This clause contains informative text only.

1449 The following use cases and object diagrams illustrate use of this profile. They are for informative
1450 purposes only and do not introduce behavioral requirements for implementations of the profile.

1451 **9.1 Instance diagram**

1452 Figure 5 depicts the CIM representation of a host system with one storage resource pool and one virtual
1453 system. Only information relevant in the context of storage resource virtualization is shown.



1454

1455

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

- 1456 In Figure 5 the host system is represented by an instance HOST of the CIM_System class. The host
1457 system owns or has access to two storage volumes each with a size of 1TB that are represented by the
1458 CIM_StorageExtent instances HVOL1 and HVOL2. Note that the storage volumes may be located within
1459 a storage area network that is not part of the host system itself.
- 1460 The host system hosts a primordial storage resource pool that is represented by the CIM_ResourcePool
1461 instance STOR_POOL. The value of the ResourceType property in STOR_POOL is 31 (Logical Disk),
1462 designating the type of resources that are allocated out of the resource pool.
- 1463 The resource type of resources aggregated by a resource pool may be different from the type of
1464 resources allocated from the pool. In this example as shown in Figure 5 both host storage volumes are
1465 aggregated into the pool, as represented by CIM_Component instances connecting STOR_POOL with
1466 HVOL1 and HVOL2, respectively.
- 1467 In the example shown in Figure 5 the storage allocation capabilities of the host system and of the storage
1468 resource pool are identical and represented by the same CIM_AllocationCapabilities instance CAP. Four
1469 SASD instances (CAP_DEF, CAP_MIN, CAP_MAX, and CAP_INC) are associated with CAP through
1470 CIM_SettingsDefineCapabilities (SDC) instances. Not shown Figure 5 are the values of the ValueRange
1471 and ValueRole properties in the SDS instances that designate the referenced SASD instances as
1472 representing the default, minimum, maximum, and increment for storage resource allocations that are
1473 supported by the system and the pool; instead the lines depicting the association instances are
1474 respectively labeled as SDC(DEF, SDC(MIN), SDC(INC) and (SDC(MAX). The values of the
1475 VirtualQuantity property in the CAP_xxx instances indicate that virtual disks allocatable from the resource
1476 pool have a minimum supported size of 1 MB up to a maximum supported size of 4TB, and that an
1477 increment of 1 MB applies within the supported range; the default size is 256 GB.
- 1478 In the CAP_xxx instances the value of the AllocationUnit property is "count"; this indicates that the value
1479 of the Reservation property is expressed in blocks. The block size is exposed by the value of the
1480 HostResourceBlockSize property (1024). Consequently storage allocations as seen from the providing
1481 host system or resource pool side are expressed in 1-KB blocks.
- 1482 Similarly, the value of the VirtualQuantityUnits property is "count", indicating that the value of the
1483 VirtualQuantity property is expressed in blocks. Here the block size is exposed by the value of the
1484 VirtualResourceBlockSize property (1024). Consequently storage allocations as seen by the consuming
1485 virtual system are expressed in 1-KB blocks as well.
- 1486 Note that the CAP_xxx instances do not expose a value for the Limit property. This indicates that the
1487 implementation does not support thin provisioning where the resource on the consuming side appears
1488 larger than the amount of resource provided at the providing side. This implies that the values of the
1489 numeric properties Reservation and VirtualQuantity are always identical for any resource allocation out of
1490 the resource pool.
- 1491 The host system hosts a virtual system that is represented by the CIM_ComputerSystem instance VS.
1492 The hosted relationship is shown through a CIM_HostedDependency instance.
- 1493 The head element of the "state" virtual system configuration is the VSSD instance STA_VSSD; it is
1494 associated with VS through a CIM_SettingsDefineState (SDS) instance. The "State" virtual system
1495 configuration contains the SASD instance DISK_STA that represents a storage resource allocation
1496 assigned to the virtual system. The virtual disk that is the result of the storage resource allocation is
1497 represented as part of the virtual system representation by the CIM_LogicalDisk instance VS_DISK.
- 1498 NOTE All instances in Figure 5 that are marked with light yellow color represent "State" entities that exist only as
1499 long as the virtual system is instantiated (that is, in a state other than "Defined"). These instances do not
1500 exist while the virtual system is not instantiated (that is, in the "Defined" state).
- 1501 The head element of the "defined" virtual system configuration is the VSSD instance DEF_VSSD; it is
1502 associated with the head element of the "state" virtual system configuration through an instance of the
1503 CIM_ElementSettingData association where the value of the IsDefault property is 1 (Is Default)
1504 (abbreviated as ESD(D) in Figure 5). The "defined" virtual system configuration contains the SASD

1505 instance DISK_DEF that represents the respective storage resource allocation request. When the virtual
1506 system is activated, respective storage resources are allocated based on their definition.

1507 Similarly to the representation of the allocation capabilities of a resource pool or system, the mutability of
1508 both the storage resource allocation request in the "Defined" virtual system configuration and of the
1509 storage resource allocation in the "State" virtual system configuration is represented by
1510 CIM_AllocationCapabilities instances with associated SASD instances through parameterized
1511 CIM_SettingsDefineCapabilities instances designating the minimum, maximum, and increment for storage
1512 resource allocation changes.

1513 Acceptable virtual system states for the removal of virtual disks are different for the storage resource
1514 allocation request and the storage resource allocation. The storage resource allocation can be removed
1515 only while the virtual system remains instantiated, as indicated by a value of 2 (Enabled) in the
1516 CIM_AllocationCapabilities instance STA_MUT. This is a manifestation of the previously mentioned fact
1517 that the "state" configuration is not present while the virtual system is in the "defined" state.

1518 9.2 Inspection

1519 This set of use cases describes how to obtain various CIM instances that represent storage-related
1520 information of host and virtual systems.

1521 9.2.1 Inspect the set of virtual disks of an active virtual system

1522 **Preconditions:** All of the following:

- 1523 • The client knows a reference to the CIM_ComputerSystem instance that represents the active
1524 virtual system.

1525 **Flow of activities:**

- 1526 1) From the CIM_ComputerSystem instance the client resolves the CIM_SystemDevice
1527 association to find the CIM_LogicalDisk instances that represent virtual disks.
- 1528 2) For each element of the result set of step 1) the client applies the use case in 9.2.2.

1529 **Postconditions:** The client knows the virtual disks of the virtual system and their properties.

1530 9.2.2 Inspect the properties of a virtual disk

1531 **Preconditions:** All of the following:

- 1532 • The client knows a reference to the instance of the CIM_LogicalDisk class that represents the
1533 virtual disk.

1534 **Flow of activities:**

- 1535 • The client obtains the CIM_LogicalDisk instance, using the GetInstance() intrinsic operation. In
1536 that instance, the client interprets property values such as the following:
 - 1537 – The value of the BlockSize property conveys the block size in effect for the virtual disks
 - 1538 – The value of the NumberOfBlocks property conveys the size of the virtual disk as seen by
1539 the virtual system as a number of blocks
 - 1540 – The value of the Name property conveys the name of the virtual disk as seen by the virtual
1541 system

1542 **Postconditions:** The client knows properties of the virtual disk.

1543 9.2.3 Determine the allocation capabilities or allocation mutability

1544 This use case is applicable in two cases:

- 1545 • Case (A) – Determine the capabilities of a system or a resource pool: In this case the entry
1546 element is the CIM_System instance representing the host system or the CIM_ResourcePool
1547 instance representing the resource pool.
- 1548 • Case (B) – Determine the mutability of a resource allocation request or resource allocation: In
1549 this case the entry element is the RASD or SASD instance representing the resource allocation
1550 request or the resource allocation.

1551 **Preconditions:** The client knows the instance path of the entry element.

1552 **Flow of activities:**

- 1553 1) The client invokes the AssociatorsNames() intrinsic operation from the entry element through
1554 the CIM_ElementCapabilities association to obtain the set of instance paths to those
1555 CIM_AllocationCapabilities instances that represent the allocation capabilities (case (A)) or
1556 mutability (case (B)) of the entry element.
- 1557 2) For each instance path obtained in step 1) the client invokes the References() intrinsic
1558 operation to obtain the set of instances of the CIM_SettingsDefineCapabilities association that
1559 reference each CIM_AllocationCapabilities instance from step 1).
- 1560 3) For each CIM_SettingsDefineCapabilities instance obtained in step 2) the client inspects the
1561 values of the ValueRole and ValueRange properties; these values define the type of limitation
1562 imposed by the RASD instance that is referenced by the value of the PartComponent property
1563 in the CIM_SettingsDefineCapabilities association instance, as follows:
 - 1564 – A default setting is designated through a value of 0 (Default) for the ValueRole property
1565 and a value of 0 (Point) for the ValueRange property. A default setting does not apply for
1566 the description of mutability.
 - 1567 – A minimum setting is designated through a value of 3 (Supported) for the ValueRole
1568 property and a value of 1 (Minimums) for the ValueRange property.
 - 1569 – A maximum setting is designated through a value of 3 (Supported) for the ValueRole
1570 property and a value of 2 (Maximums) for the ValueRange property.
 - 1571 – An increment setting is designated through a value of 3 (Supported) for the ValueRole
1572 property and a value of 3 (Increments) for the ValueRange property.
- 1573 4) For each of the CIM_SettingsDefineCapabilities association instances obtained in step 2) and
1574 inspected in step 3) the client invokes the intrinsic GetInstance() CIM operation, using the value
1575 of the PartComponent property as input for the InstanceName parameter. The result each time
1576 is a RASD instance where values of all non-null numeric properties describe the settings in the
1577 context established by the CIM_SettingsDefineState instance inspected in step 3).

1578 **Postconditions:** The client knows the allocation capabilities of the system or the resource pool
1579 (case (A)), or the mutability of a resource allocation request or a resource allocation (case (B)).

1580 9.2.4 Determine the default resource allocation capabilities

1581 **Preconditions:** The client knows all of the following:

- 1582 • A reference to the CIM_System instance that represents the host system.
- 1583 • A selected resource type (such as for example 31 (Logical Disk) or 17 (Disk Drive))

1584 **Flow of activities:**

1585 1) The client obtains instances of the CIM_ElementCapabilities association that reference the
1586 instance of the CIM_System class, invoking the References() intrinsic operation with parameter
1587 values set as follows:

1588 – The value of the ObjectName parameter refers to the instance of the CIM_System class.

1589 – The value of the ResultClass parameter is set to "CIM_ElementCapabilities".

1590 The result of step 1) is a set of instances of the CIM_ElementCapabilities association.

1591 2) From the result set of step 1), the client drops those instances where the value set of the
1592 Characteristics[] array property does not contain an element with the value 2 (Default).

1593 The result of this step is a set of instances of the CIM_ElementCapabilities association that
1594 reference CIM_AllocationCapabilities instances that represent the default allocation capabilities
1595 of the system for a number of resource types.

1596 3) For each of the association instances obtained in step 2), the client obtains the
1597 CIM_AllocationCapabilities instance that is referenced by the value of the Capabilities property
1598 in the respective association instance, invoking the intrinsic GetInstance() CIM operation with
1599 the value of the InstanceName parameter set to the value of the Capabilities property.

1600 The result of step 3) is a set of CIM_AllocationCapabilities instances that represent the system's
1601 default allocation capabilities for a number of resource types.

1602 4) From the result set of step 3), the client drops those instances where the value set of the
1603 ResourceType property does not match the selected resource type.

1604 The result of this step is one RASD instance that represents the system's default allocation
1605 capabilities for the selected resource type. The client continues as in use case 9.2.3 step 2) in
1606 order to determine the set of RASD instances that represent the settings for the default
1607 resource allocation capabilities for the selected resource type.

1608 **Postconditions:** The client knows the default allocation capabilities of the system for the selected
1609 resource type.

1610 In the example CIM representation shown in Figure 5, the default allocation capabilities for the storage
1611 extent resource type of the system are represented by the CIM_AllocationCapabilities instance CAP and
1612 related RASD instances.

1613 **9.2.5 Determine the default resource pool**

1614 **Preconditions:** The client knows a reference to the CIM_AllocationCapabilities instance that represents
1615 the default resource allocation capabilities of the system for a selected resource type; see 9.2.4.

1616 **Flow of activities:**

1617 1) The client obtains instances of the CIM_ElementCapabilities association that reference the
1618 instance of the CIM_AllocationCapabilities class, invoking the intrinsic References() CIM
1619 operation with parameter values set as follows:

1620 – The value of the ObjectName parameter refers to the CIM_AllocationCapabilities instance.

1621 – The value of the ResultClass parameter is set to "CIM_ElementCapabilities".

1622 The result of this step is a set of instances of the CIM_ElementCapabilities association.

1623 2) From the result set of step 1), the client drops those instances where the value set of the
1624 Characteristics[] array property does not contain an element with the value 2 (Default).

1625 The result of this step is a set of two instances of the CIM_ElementCapabilities association. One
 1626 association instance references the CIM_ResourcePool instances that represent the default
 1627 resource pool, and one instance references the CIM_System instance that represents the host
 1628 system.

1629 3) The client selects the instance of the CIM_ElementCapabilities association from the result of
 1630 step 2) that references the CIM_ResourcePool instance by comparing the value of the
 1631 ManagedElement property against the known reference to the CIM_System instance that
 1632 represents the host system and dropping that association instance. The client uses the
 1633 remaining association instance from the result set of step 2) to obtain the CIM_ResourcePool
 1634 instance that is referenced by the value of the ManagedElement property in that association
 1635 instance, invoking the intrinsic GetInstance() CIM operation with the value of the InstanceName
 1636 parameter set to the value of the ManagedElement property.

1637 The result of this step is the CIM_ResourcePool instance that represents the system's default
 1638 resource pool for the selected resource type.

1639 **Postconditions:** The client knows the default resource pool of the system for the selected resource type.

1640 In the example CIM representation shown in Figure 5, the default storage resource pool is represented by
 1641 the CIM_ResourcePool instance STOR_POOL.

1642 9.2.6 Obtain the storage resource pool with the largest unreserved capacity

1643 **Preconditions:**

- 1644 • The client knows a reference to the CIM_System instance that represents the host system.

1645 **Flow of activities:**

1646 1) The client resolves the CIM_HostedPool association to find the CIM_ResourcePool instances
 1647 that represent resource pools hosted by the host system, invoking the intrinsic
 1648 AssociatorNames() CIM operation with parameter values set as follows:

- 1649 – The value of the ObjectName parameter refers to the CIM_System instance that
 1650 represents the host.
- 1651 – The value of the AssocClass parameter is set to "CIM_HostedPool".
- 1652 – The value of the ResultClass parameter is set to "CIM_ResourcePool".

1653 The result of this step is a set of CIM_ResourcePool instances that represent resource pools
 1654 hosted by the host system.

1655 2) The client selects from the result set of step 1) only those instances where the value of the
 1656 ResourceType property matches 31 (Logical Disk).

1657 The result is a set of CIM_ResourcePool instances that represent storage resource pools
 1658 hosted by the host system.

1659 3) The client inspects the value of the Capacity and the Reserved properties in all instances
 1660 selected with step 2), and each time calculates the amount of unreserved storage capacity by
 1661 subtracting the value of the Reserved property from the value of the Capacity property.

1662 4) From all pools inspected in step 3), the client selects the one that has the largest free capacity.

1663 5) The client checks the resource pool selected in step 4) for architectural limitations as expressed
 1664 by the pool's capabilities, applying use case 9.2.3.

1665 **Postconditions:** The client knows the resource pool with the largest unreserved storage capacity.

1666 In the example CIM representation shown in Figure 5, the client initially would know the CIM_System
 1667 instance HOST that represents the host system. From there, the client would follow the CIM_HostedPool

1668 association to locate CIM_ResourcePool instances. Typically the association resolution would yield more
 1669 than one instance, including instances that represent resource pools of other resource types;
 1670 consequently the client is required to select only those instances where the value of the ResourceType
 1671 property matches 31 (Logical Disk). In Figure 5, there is only one CIM_ResourcePool instance in the
 1672 result set that is named STOR_POOL. From that instance, the client takes the value of the Capacity
 1673 property and subtracts the value of the Reserved property (2147483648 – 268435456) byte, yielding
 1674 1879048192 blocks (or 1792 GB) as the maximum storage capacity presently available from the pool.

1675 9.3 Management

1676 This set of use cases describes how to create new virtual disks, and how to modify existing virtual disks.
 1677 These management tasks are described in terms of a virtual system management service, as represented
 1678 by a CIM_VirtualSystemManagementService instance.

1679 9.3.1 Create virtual disk (block based)

1680 **Preconditions:** All of the following:

- 1681 • The client knows a reference to the CIM_ComputerSystem instance that represents the virtual
1682 system.
- 1683 • The client knows a reference to the CIM_VirtualSystemManagementService instance that
1684 represents the virtual system management service responsible for the virtual system.
- 1685 • The client has performed the use case and knows the default allocation capabilities of the
1686 system.
- 1687 • The size of the new disk is 256 GB (or 268435456 blocks with a size of 1 KB (1024 bytes)).

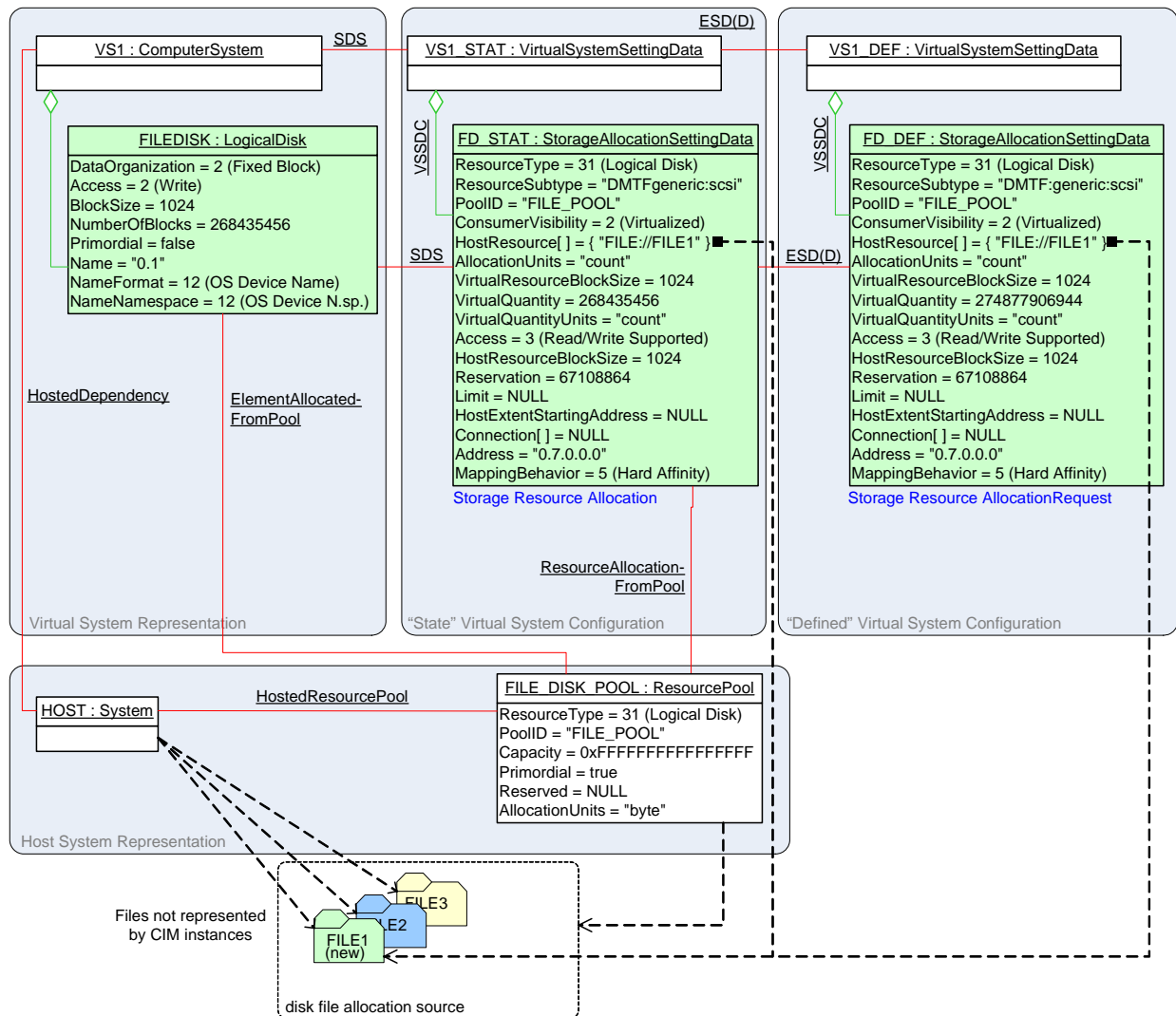
1688 **Flow of activities:**

- 1689 1) The client locally prepares a SASD instance, with properties set as follows:
 - 1690 – ResourceType: 31 (Logical Disk) // device type as seen by consumer
 - 1691 – ResourceSubtype: // implementation dependent
 - 1692 – PoolID: NULL // implies default pool
 - 1693 – AllocationUnits: "count" // count of blocks; if value is NULL, the effective value
1694 // is implied by pool capabilities
 - 1695 – VirtualQuantity: 268435456 // 256 GB
 - 1696 – VirtualQuantityBlockSize: 1024 // may be NULL, implied by pool capabilities
 - 1697 – VirtualQuantityUnits: "count" // count of blocks; if value is NULL, the effective value
1698 // is implied by pool capabilities
 - 1699 – Reservation: NULL // may be NULL if thin provisioning is not requested;
1700 // defaults to the size expressed by the value of the
1701 // VirtualQuantity property
 - 1702 – Limit: NULL // defaults to maximum disk size as expressed by the
1703 // value of the VirtualQuantity property
 - 1704 – Address: "/dev/sda1" // optional; if not specified the implementation
1705 // assigns an address
 - 1706 – Values of all other properties are not set (NULL), requesting a default behavior
- 1707 2) The client invokes the AddResourceSettings() method of the virtual system management
1708 service, with parameters set as follows:

- 1748 – It is assumed that the method returns 0, indicating successful synchronous execution.
- 1749 – As a side effect, the new file FILE1 is created in a default directory. The initial size of the
- 1750 file is 64 GB, up to a limit of 256 GB. The disk size as seen by the virtual system is 256 GB
- 1751 from the beginning.

1752 **Postconditions:** A new file-based virtual disk is created for the virtual system, as requested.

1753 Figure 6 shows the situation that results after the create disk operation completed and the virtual system
 1754 was activated:



1755

1756

Figure 6 – Create virtual disk with implicit file creation

1757 9.3.3 Create virtual disk (file based pre-existing)

1758 **Preconditions:** All of the following:

- 1759 • The client knows a reference to the VSSD instance that represents the virtual system
 1760 configuration to receive the new virtual disk.

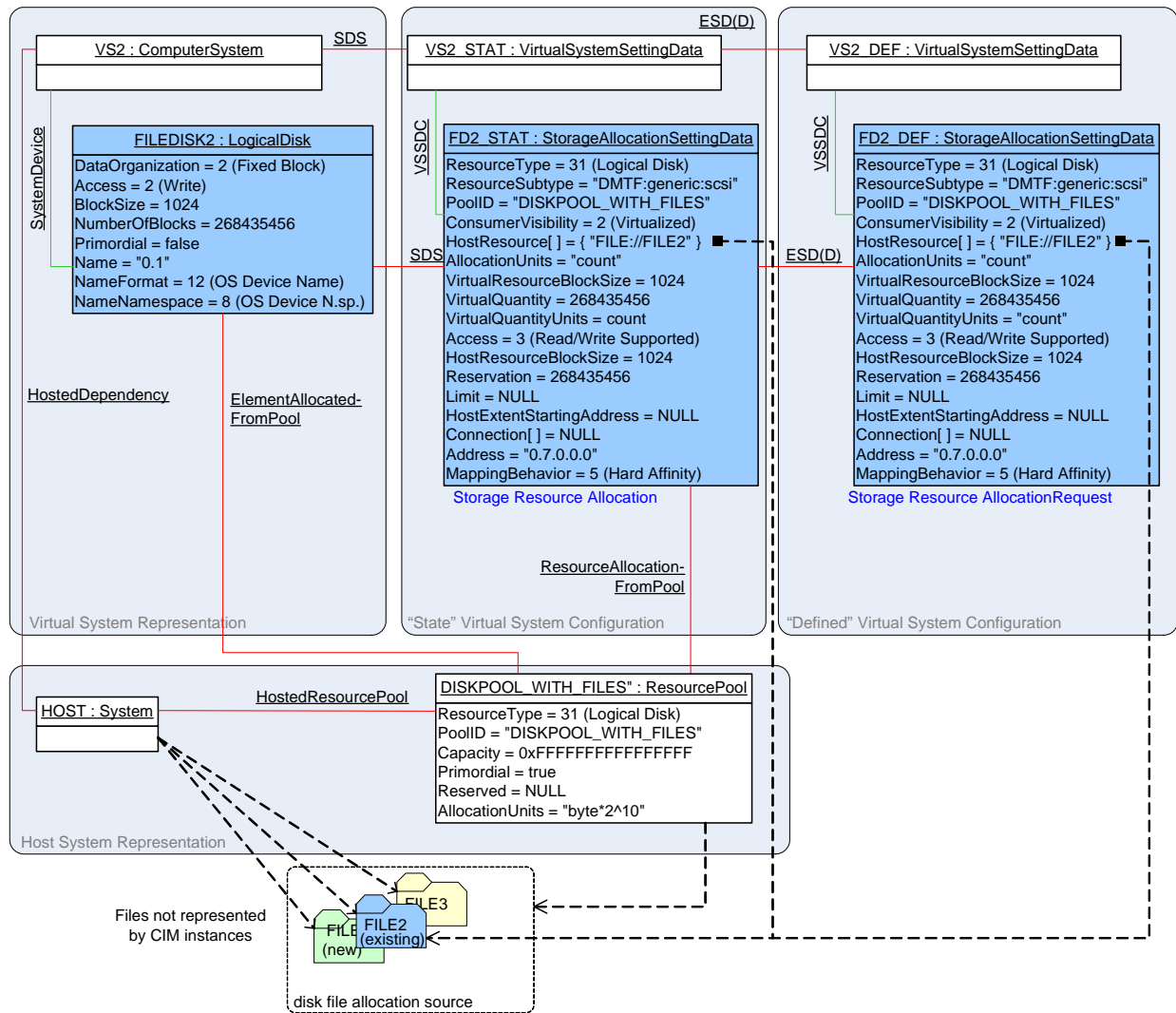
- 1761 • The client knows a reference to the CIM_VirtualSystemManagementService instance that
1762 represents the virtual system management service responsible for the virtual system.
- 1763 • The client knows the URI "FILE://FILE2" of a pre-existing file that contains the data for the new
1764 disk.
- 1765 • The size of the new disk is implied by the content stored in the file.

1766 **Flow of activities:**

- 1767 1) The client locally prepares a SASD instance, with properties set as follows:
- 1768 – ResourceType: 31 (Logical Disk)
 - 1769 – ResourceSubtype: "DMTF:generic:scsi" // Microsoft SCSI disk
 - 1770 – PoolID: "FILE_DISK_POOL" // implementation specific
1771 // dummy pool
 - 1772 – AllocationUnits: "count" // number of blocks
 - 1773 – VirtualQuantity: NULL // disk size implied by file
 - 1774 – Reservation: NULL // disk reservation implied by file
 - 1775 – HostResource[0]: "FILE://FILE2" // URI referring to pre-existing file
 - 1776 – Address: "0.7.0.0.0" // SCSI bus 0 / initiator 7 / target 0 / lun 0
1777 // partition 0
 - 1778 – MappingBehavior: 5 (Hard Affinity) // implies that the virtual system
1779 // requires this disk at startup
 - 1780 – Values of all other properties are not set (NULL), requesting a default
- 1781 2) The client invokes the AddResourceSettings() method of the virtual system management
1782 service, with parameters set as follows:
- 1783 – AffectedConfiguration: REF to the VSSD instance that represents
1784 the "Defined" virtual system configuration.
 - 1785 – ResourceSettings: One element with the embedded instance
1786 prepared in step 1)
- 1787 The implementation executes the AddResourceSettings() method.
- 1788 – It is assumed that the method returns 0, indicating successful synchronous execution.
 - 1789 – The new disk is configured into the virtual system configuration. It is based on the file
1790 provided as input. The initial disk size is 268435456 KB, as implied by the disk content
1791 stored in the file.

1792 **Postconditions:** A new file-based virtual disk is created for the virtual system, as requested.

1793 Figure 7 shows the situation that results after the create disk operation completed and the virtual system
1794 was activated:



1795

1796

Figure 7 – Create virtual disk with pre-existing file

1797 **9.3.4 Create virtual disk (block based passed-through)**

1798 **Preconditions:** All of the following:

- 1799 • The client knows a reference to the VSSD instance that represents the virtual system
- 1800 configuration to receive the new dedicated virtual disk.
- 1801 • The client knows a reference to the CIM_VirtualSystemManagementService instance that
- 1802 represents the virtual system management service responsible for the virtual system.
- 1803 • The size of the new disk is 256 GB (or 268435456 kbyte).

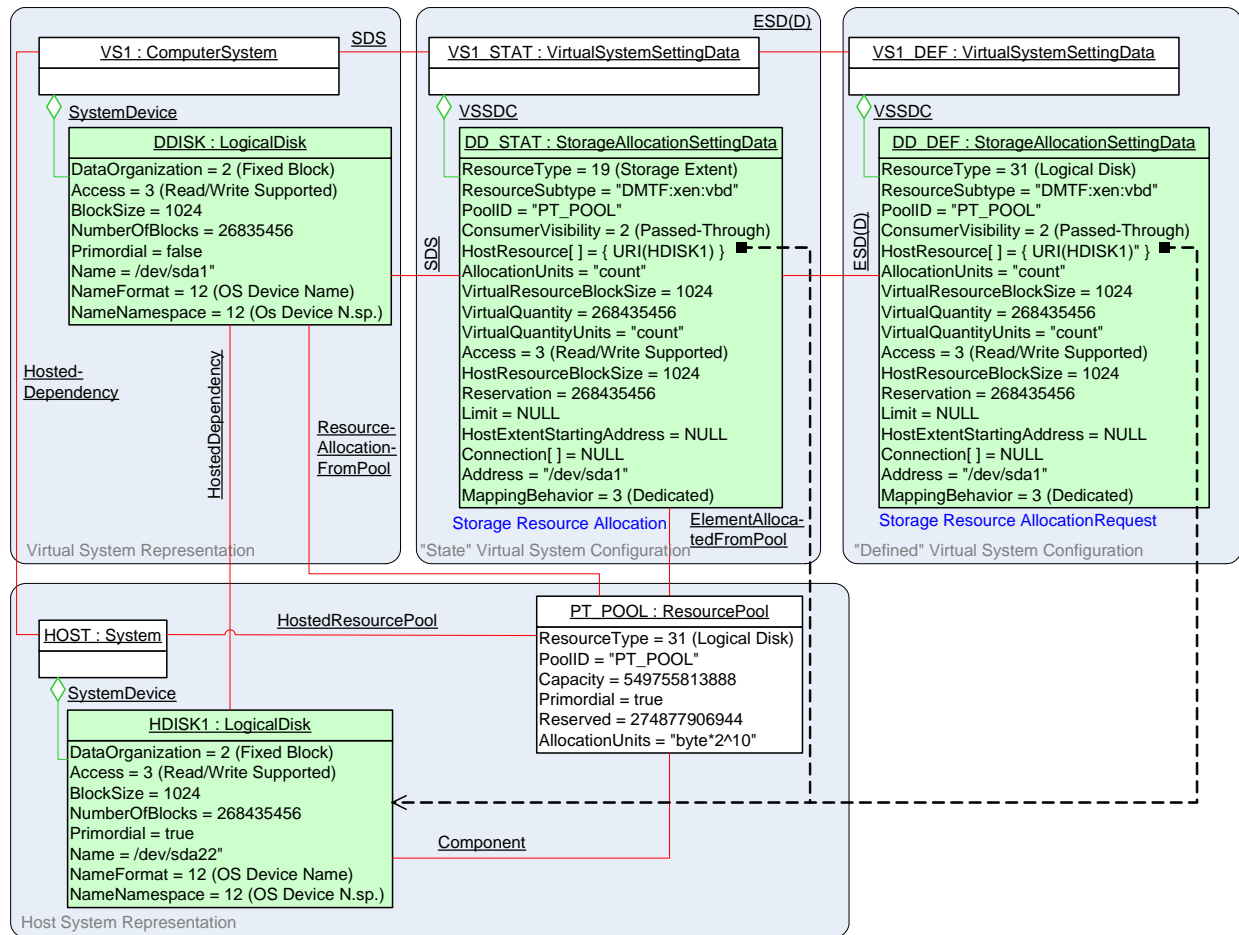
1804 **Flow of activities:**

- 1805 1) The client locally prepares an instance of the CIM_ResourceAllocationSettingData class, with
- 1806 properties set as follows:
- 1807 – ResourceType: 31 (Logical Disk)
- 1808 – ResourceSubtype: "DMTF:xen:vbd"


```

1809     - PoolID:           "PT_POOL"           // Example; refers to a pool with disks
1810                                     // that are passed-through
1811     - AllocationUnits:   "count"             // number of blocks
1812     - VirtualQuantity:   268435456          // 256 GB; needed if not explicit host resource
1813                                     // requested
1814     - VirtualResourceBlockSize: 1024        // blocksize as seen by consumer
1815     - VirtualQuantityUnits: "count"         // number of blocks
1816     - Reservation:       274877906944      // needed if no explicit
1817                                     // host resource requested
1818     - HostResource[0]:   "URI(HDISK1)"      // optional; may refer to a
1819                                     // specific disk in the pool; if not
1820                                     // specified, the implementation
1821                                     // selects a disk out of the pool
1822     - Address:           "/dev/sda1"        // optional; if not specified the
1823                                     // implementation assigns an
1824                                     // address
1825     - Values of all other properties are not set (or are NULL), requesting a default
1826 2) The client invokes the AddResourceSettings( ) method of the virtual system management
1827    service, with parameters set as follows:
1828     - AffectedConfiguration: REF to CIM_VirtualSystemSettingData instance
1829     - ResourceSettings:      One element with the embedded instance
1830                               prepared in step 1)
1831
1831     It is assumed that the method returns 0, indicating successful synchronous execution.
1832 Postconditions: A new passed-through virtual disk is created for the virtual system, as requested.
1833 Figure 8 shows the situation that results after the AddResourceSettings( ) operation completes.

```



1835 **Figure 8 – Create dedicated virtual disk**

1836 In this example the resource pool represented by PT_POOL aggregates a set of host disks that were set
 1837 aside for the purpose of being passed-through to virtual systems. Adding a resource from that pool is
 1838 actually a selection based upon client requirements.

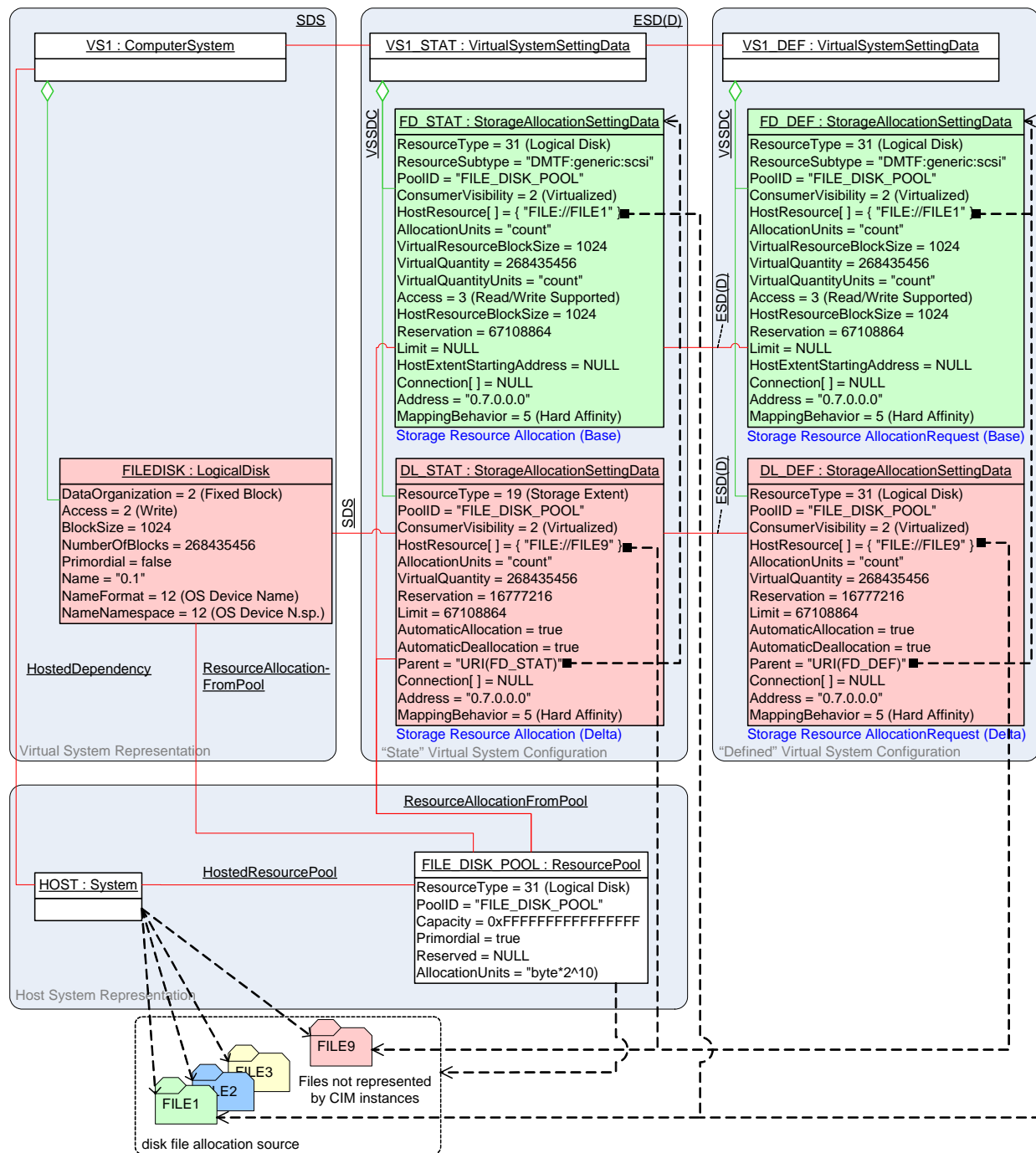
1839 9.3.5 Create virtual disk (file based delta)

1840 **Preconditions:** All of the following:

- 1841 • The situation that was the result of the use case described in 9.3.2.
- 1842 • The size of the virtual remains disk is 256 GB (or 268435456 KB).
- 1843 • The initial size of the new delta disk is 16 GB (or 16777216 KB).
- 1844 • The name of the file is "FILE9" (relative file path); the file does not exist.
- 1845 • The file is only allocated as the virtual system is activated (instantiated).
- 1846 • The file is deallocated as the virtual system is deactivated.

1847 **Flow of activities:**

- 1848 1) The client locally prepares a SASD instance with properties set as follows:
 - 1849 – ResourceType: 31 (Logical Disk)



1875

1876

Figure 9 – Create virtual delta disk and file

1877 Note that the instances `FD_STAT`, `DL_STAT` and `FILEDISK` are present only while the virtual system is
 1878 instantiated and the virtual disk is allocated. Note that there is only one disk `FILEDISK` in the virtual
 1879 system representation that is allocated based on both `FD_STAT` and `DL_STAT`. There is no separate
 1880 instance of `CIM_LogicalDisk` representing each allocation separately as there is only one virtual disk
 1881 presented to the virtual system.

1882 Note that the file `FILE9` containing the delta disk is automatically allocated during virtual disk allocation
 1883 because the value of the `AutomaticAllocation` property is true; the file is automatically deallocated during

1884 virtual disk deallocation because the value of the AutomaticDeallocation property is true. As a
 1885 consequence the virtual system at startup time receives a virtual disk that is initially based on FILE1; as
 1886 the virtual system writes onto the disk the delta is maintained in FILE9. The size of FILE9 is driven by the
 1887 values of the Reservation and the Limit properties in DL_STAT: The initial file size is 16 GB, up to a limit
 1888 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
 1889 VirtualQuantity property). That disk is initially based on the read-only file-based extent as allocated by
 1890 FD_STAT. On top of the read-only extent is a temporary delta read-write extent as allocated by DL_STAT
 1891 that enables overwriting data up to an amount of 64 GB; the delta extent is discarded when the virtual
 1892 disk is deallocated, such that the next allocation starts with the initial read-only content again.

1893 **10 CIM Elements**

1894 Table 5 lists CIM elements that are defined or specialized for this profile. Each CIM element shall be
 1895 implemented as described in Table 5. The CIM Schema descriptions for any referenced element and its
 1896 sub-elements apply.

1897 Clauses 7 ("Implementation") and 8 ("Methods") may impose additional requirements on these elements;
 1898 in particular, clause 7 ("Implementation") may impose requirements for CIM instances.

1899 **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		

1900 10.1 CIM_Component for resource pool

1901 The implementation of the CIM_Component association for the representation of the aggregation of host
1902 resources into resource pools is conditional.

1903 Condition: The resource aggregation feature (see 7.5) is implemented.

1904 The CIM_Component association is abstract; therefore it cannot be directly implemented. For this reason
1905 the provisions in this subclause shall be applied to implementations of subclasses of the CIM_Component
1906 association. However, note that clients may directly resolve abstract associations without knowledge of
1907 the concrete subclass that is implemented.

1908 Table 6 lists the requirements for elements of this association. These requirements are in addition to
1909 those specified in the CIM Schema and in [DMTF DSP1041:1.1](#).

1910 **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: *

1911 **10.2 CIM_DiskDrive for host disk drives**

1912 The implementation of the CIM_DiskDrive class for the representation of host disk drives is conditional.

1913 Condition: The resource aggregation feature is implemented for disk drive resource pools; see 7.5.

1914 Table 7 lists the requirements for elements of this class.

1915 **Table 7 – Class: CIM_DiskDrive (Host)**

Elements	Requirement	Notes
DefaultBlockSize	Mandatory	See CIM schema description

1916 **10.3 CIM_DiskDrive for virtual disk drives**

1917 The implementation of the CIM_DiskDrive class for the representation of virtual disk drives is conditional.

1918 Condition: This profile is implemented for the allocation of disk drives; see 7.2.

1919 Table 8 lists the requirements for elements of this class.

1920 **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

1921 **10.4 CIM_ElementAllocatedFromPool for allocated virtual resources**

1922 Table 9 lists the requirements for elements of this association. These requirements are in addition to
 1923 those specified in the CIM Schema and in [DMTF DSP1041:1.1](#).

1924 **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: *

1925 **10.5 CIM_ElementAllocatedFromPool for resource pool hierarchies**

1926 The implementation of the CIM_ElementAllocatedFromPool association for the representation of resource
 1927 pool hierarchies is conditional.

1928 Condition: The resource pool management feature (see 7.7) is implemented.

1929 Table 10 lists the requirements for elements of this association. These requirements are in addition to
 1930 those specified in the CIM Schema and in [DMTF DSP1041:1.1](#).

1931 **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: *

1932 10.6 CIM_ElementSettingData for resource allocation request

1933 Table 11 lists the requirements for elements of this class. These requirements are in addition to those
 1934 specified in the CIM Schema and in [DMTF DSP1041:1.1](#).

1935 **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).

1936 10.7 CIM_ElementSettingData for resource pool

1937 Table 12 lists the requirements for elements of this class. These requirements are in addition to those
 1938 specified in the CIM Schema and in [DMTF DSP1041:1.1](#).

1939 **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

1940 **10.8 CIM_HostedDependency**

1941 The implementation of the CIM_HostedDependency association is optional.

1942 Table 13 lists the requirements for elements of this association. These requirements are in addition to
 1943 those specified in the CIM Schema and in [DMTF DSP1041:1.1](#) (*Resource Allocation Profile*).

1944 **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

1945 **10.9 CIM_LogicalDisk for virtual disk**

1946 The implementation of the CIM_LogicalDisk class for the representation of virtual disks is conditional.

1947 Condition: This profile is implemented for the allocation of storage extents; see 7.2.

1948 Table 14 lists the requirements for elements of this class in addition to those specified for the
 1949 implementation of the CIM_StorageExtent class for the representation of virtual disks; see 10.17.

1950 **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.

1951 **10.10 CIM_ReferencedProfile**

1952 Table 15 lists the requirements for elements of this association. These requirements are in addition to
 1953 those specified in the CIM Schema and in [DMTF DSP1033:1.0](#) (*Profile Registration Profile*).

1954 **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

1955 10.11 CIM_RegisteredProfile

1956 Table 16 lists the requirements for elements of this class. These requirements are in addition to those
1957 specified in the CIM schema and in [DMTF DSP1033:1.0 \(Profile Registration Profile\)](#).

1958 **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".

1959 10.12 CIM_ResourceAllocationSettingData for disk drive allocation information

1960 The implementation of the CIM_ResourceAllocationSettingData class for the representation of disk drive
1961 allocation information is conditional.

1962 Condition: This profile is implemented for the allocation of disk drives; see 7.2.

1963 Table 17 lists the requirements for elements of this class. These requirements are in addition to those
1964 specified in the CIM Schema and in [DMTF DSP1041:1.1 \(Resource Allocation Profile\)](#).

1965 **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.

1966 **10.13 CIM_ResourcePool**

1967 Table 18 lists the requirements for elements of this class. These requirements are in addition to those
 1968 specified in the CIM Schema and in [DMTF DSP1041:1.1](#) (*Resource Allocation Profile*).

1969 **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.

1970 **10.14 CIM_SettingsDefineState**

1971 Table 19 lists the requirements for elements of this association. These requirements are in addition to
 1972 those specified in the CIM Schema and in [DMTF DSP1041:1.1](#) (*Resource Allocation Profile*).

1973

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

1974 **10.15 CIM_StorageAllocationSettingData for storage allocation information**

1975 The implementation of the CIM_StorageAllocationSettingData class for the representation of storage
1976 allocation information is conditional.

1977 Condition: This profile is implemented for the allocation of storage extents; see 7.2.

1978 Table 20 lists the requirements for elements of this class. These requirements are in addition to those
1979 specified in the CIM Schema and in [DMTF DSP1041:1.1](#) (*Resource Allocation Profile*).

1980

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.

1981 **10.16 CIM_StorageVolume for host storage volume**

1982 The implementation of the CIM_StorageVolume class for the representation of host storage volumes is
 1983 conditional.

1984 Condition: The storage resource aggregation feature is implemented; see 7.5.

1985 Table 21 lists the requirements for elements of this class.

1986 **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.

1987 **10.17 CIM_StorageExtent for virtual storage extent**

1988 See 7.9 for detailed implementation requirements for this class if it is used for the representation of virtual
 1989 disks.

1990 Table 22 lists the requirements for elements of this class.

1991 **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.

1992 **10.18 CIM_SystemDevice for host storage volumes**

1993 The implementation of the CIM_SystemDevice association for host storage volumes is conditional.

1994 Condition: The storage resource aggregation feature is implemented; see 7.5.

1995 Table 23 lists the requirements for elements of this association.

1996 **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: *

1997 **10.19 CIM_SystemDevice for virtual resources**

1998 Table 24 lists the requirements for elements of this association.

1999 **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: *

2000

**ANNEX A
(Informative)****Change Log**2001
2002
2003
2004

2005

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

2006