



1

2

3

4

**Document Number: DSP1045**

**Date: 2009-07-14**

**Version: 1.0.0**

5

# **Memory Resource Virtualization Profile**

6

**Document Type: Specification**

7

**Document Status: DMTF Standard**

8

**Document Language: E**

9

## 10 Copyright Notice

11 Copyright © 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

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

# CONTENTS

33	Foreword .....	5
34	Introduction .....	6
35	1 Scope .....	7
36	2 Normative References.....	7
37	3 Terms and Definitions .....	7
38	4 Symbols and Abbreviated Terms .....	10
39	5 Synopsis.....	11
40	6 Description (Informative) .....	11
41	6.1 General .....	11
42	6.2 Memory Resource Virtualization Class Schema .....	12
43	6.3 Memory Resource Pool .....	13
44	6.4 Memory Resource Allocation .....	14
45	7 Implementation.....	18
46	7.1 Allocation Units .....	18
47	7.2 Resource Allocation Profile.....	19
48	7.3 Allocation Capabilities Profile .....	26
49	7.4 System Memory Profile.....	29
50	8 Methods.....	29
51	8.1 General .....	29
52	8.2 Profile conventions for operations .....	30
53	8.3 CIM_RegisteredProfile.....	30
54	9 Use Cases (Informative).....	30
55	9.1 Object Diagram .....	30
56	9.2 Inspection.....	33
57	10 CIM Elements.....	39
58	10.1 CIM_AffectedJobElement .....	40
59	10.2 CIM_AllocationCapabilities (Capabilities) .....	41
60	10.3 CIM_AllocationCapabilities (Mutability) .....	41
61	10.4 CIM_Component (Memory Composition) .....	42
62	10.5 CIM_Component (Resource Pool).....	42
63	10.6 CIM_ConcreteJob .....	43
64	10.7 CIM_ElementAllocatedFromPool.....	44
65	10.8 CIM_ElementCapabilities (Capabilities) .....	44
66	10.9 CIM_ElementCapabilities (Mutability).....	45
67	10.10 CIM_ElementSettingData (Memory Resource Pool) .....	45
68	10.11 CIM_ElementSettingData (Memory Resource) .....	46
69	10.12 CIM_HostedDependency.....	46
70	10.13 CIM_Memory (Host System) .....	47
71	10.14 CIM_Memory (Virtual System).....	47
72	10.15 CIM_RegisteredProfile.....	48
73	10.16 CIM_ResourceAllocationFromPool.....	48
74	10.17 CIM_ResourceAllocationSettingData .....	48
75	10.18 CIM_ResourcePool.....	49
76	10.19 CIM_ResourcePoolConfigurationCapabilities .....	50
77	10.20 CIM_SettingsDefineState .....	50
78	10.21 CIM_ServiceAffectsElement .....	50
79	10.22 CIM_SystemDevice (Virtual Memory).....	51
80	10.23 CIM_SystemDevice (Host Memory) .....	51
81	ANNEX A (Informative) Change Log .....	52
82		

## 83 Figures

84	Figure 1 – Memory Resource Virtualization Profile: Profile Class Diagram .....	12
85	Figure 2 – Instance Diagram: Concept of Memory Resource Pool Hierarchies .....	14
86	Figure 3 – Instance Diagram: Concept of Memory Resource Allocation .....	15
87	Figure 4 – Instance Diagram: Memory Composition .....	17
88	Figure 5 – Instance Diagram: Example CIM Representation of Memory Resource Virtualization .....	32
89		

## 90 Tables

91	Table 1 – Related Profiles .....	11
92	Table 2 – CIM Elements: Memory Resource Virtualization Profile .....	40
93	Table 3 – Association: CIM_AffectedJobElement .....	41
94	Table 4 – Class: CIM_AllocationCapabilities (Memory Allocation Capabilities) .....	41
95	Table 5 – Class: CIM_AllocationCapabilities (Memory Allocation Mutability) .....	42
96	Table 6 – Association: CIM_Component (Memory Resource) .....	42
97	Table 7 – Association: CIM_Component (Resource Pool) .....	43
98	Table 8 – Class: CIM_ConcreteJob .....	43
99	Table 9 – Association: CIM_ElementAllocatedFromPool .....	44
100	Table 10 – Association: CIM_ElementCapabilities (Capabilities) .....	44
101	Table 11 – Association: CIM_ElementCapabilities (Mutability) .....	45
102	Table 12 – Association: CIM_ElementSettingData (Memory Resource Pool) .....	45
103	Table 13 – Association: CIM_ElementSettingData (Memory Resource) .....	46
104	Table 14 – Association: CIM_HostedDependency .....	46
105	Table 15 – Class: CIM_Memory (Host System) .....	47
106	Table 16 – Class: CIM_Memory (Virtual System) .....	47
107	Table 17 – Class: CIM_RegisteredProfile .....	48
108	Table 18 – Association: CIM_ResourceAllocationFromPool .....	48
109	Table 19 – Class: CIM_ResourceAllocationSettingData .....	49
110	Table 20 – Class: CIM_ResourcePool .....	49
111	Table 21 – Class: CIM_ResourcePoolConfigurationCapabilities .....	50
112	Table 22 – Association: CIM_SettingsDefineState .....	50
113	Table 23 – Association: CIM_ServiceAffectsElement .....	50
114	Table 24 – Association: CIM_SystemDevice (Virtual Memory) .....	51
115	Table 25 – Association: CIM_SystemDevice (Host Memory) .....	51
116		

117

## Foreword

118 The *Memory Resource Virtualization Profile* (DSP1045) was prepared by the System Virtualization,  
119 Partitioning and Clustering Working Group of the DMTF.

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

122 Editors:

- 123 • Michael Johanssen – IBM
- 124 • Ron Goering – IBM

125 Contributors:

- 126 • Gareth Bestor – IBM
- 127 • Ron Goering – IBM
- 128 • Daniel Hiltgen –VMware
- 129 • Ron Doyle – IBM
- 130 • Rene Schmidt – VMware Inc.
- 131 • Steffen Gararup – VMware Inc.
- 132 • Hemal Shah – Broadcom
- 133 • Fred Maciel – Hitachi Ltd.
- 134 • Lawrence Lamers – VMware Inc.
- 135 • Andreas Maier – IBM
- 136 • John Parchem – Microsoft Corporation
- 137 • George Ericson – EMC
- 138 • Oliver Benke – IBM
- 139 • John Leung – Intel Corporation
- 140 • James Fehlig – Novell
- 141 • Nihar Shah – Microsoft Corporation
- 142 • Shishir Pardikar – Citrix Systems Inc.
- 143 • Stephen Schmidt – IBM
- 144 • Mark Hapner – Sun Microsystems
- 145 • Dave Barrett – Emulex
- 146 • John Suit – Fortisphere
- 147 • Jeff Wheeler – Cisco
- 148 • Mark Johnson – IBM
- 149 • Carl Waldsburger – VMware Inc.

150

151

## Introduction

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

157

# Memory Resource Virtualization Profile

## 158 1 Scope

159 This profile is a component DMTF management profile that extends the management capabilities of the  
160 referencing profile by adding the support to represent and manage the allocation of memory to virtual  
161 systems.

## 162 2 Normative References

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

166 DMTF DSP0004, *CIM Infrastructure Specification 2.5*,  
167 [http://www.dmtf.org/standards/published\\_documents/DSP0004\\_2.5.pdf](http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf)

168 DMTF DSP0200, *CIM Operations over HTTP 1.3*,  
169 [http://www.dmtf.org/standards/published\\_documents/DSP0200\\_1.3.pdf](http://www.dmtf.org/standards/published_documents/DSP0200_1.3.pdf)

170 DMTF DSP1001, *Management Profile Specification Usage Guide 1.0*,  
171 [http://www.dmtf.org/standards/published\\_documents/DSP1001\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1001_1.0.pdf)

172 DMTF DSP1026, *System Memory Profile 1.0*,  
173 [http://www.dmtf.org/standards/published\\_documents/DSP1026\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1026_1.0.pdf)

174 DMTF DSP1033, *Profile Registration Profile 1.0*,  
175 [http://www.dmtf.org/standards/published\\_documents/DSP1033\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1033_1.0.pdf)

176 DMTF DSP1041, *Resource Allocation Profile 1.1*,  
177 [http://www.dmtf.org/standards/published\\_documents/DSP1041\\_1.1.pdf](http://www.dmtf.org/standards/published_documents/DSP1041_1.1.pdf)

178 DMTF DSP1043, *Allocation Capabilities Profile 1.0*,  
179 [http://www.dmtf.org/standards/published\\_documents/DSP1043\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1043_1.0.pdf)

180 ISO/IEC Directives, Part 2, *Rules for the structure and drafting of International Standards*,  
181 <http://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype>

## 182 3 Terms and Definitions

183 For the purposes of this document, the following terms and definitions apply. For the purposes of this  
184 document, the terms and definitions given in [DSP1033](#) and [DSP1001](#) also apply.

### 185 3.1

#### 186 **can**

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

### 188 3.2

#### 189 **cannot**

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

- 191 **3.3**  
192 **conditional**  
193 indicates requirements strictly to be followed in order to conform to the document and from which no  
194 deviation is permitted when the specified conditions are met
- 195 **3.4**  
196 **mandatory**  
197 indicates requirements strictly to be followed in order to conform to the document and from which no  
198 deviation is permitted
- 199 **3.5**  
200 **may**  
201 indicates a course of action permissible within the limits of the document
- 202 **3.6**  
203 **need not**  
204 indicates a course of action permissible within the limits of the document
- 205 **3.7**  
206 **optional**  
207 indicates a course of action permissible within the limits of the document
- 208 **3.8**  
209 **referencing profile**  
210 indicates a profile that owns the definition of this class and can include a reference to this profile in its  
211 "Related Profiles" table
- 212 **3.9**  
213 **shall**  
214 indicates requirements strictly to be followed in order to conform to the document and from which no  
215 deviation is permitted
- 216 **3.10**  
217 **shall not**  
218 indicates requirements strictly to be followed in order to conform to the document and from which no  
219 deviation is permitted
- 220 **3.11**  
221 **should**  
222 indicates that among several possibilities, one is recommended as particularly suitable, without  
223 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 224 **3.12**  
225 **should not**  
226 indicates that a certain possibility or course of action is deprecated but not prohibited
- 227 **3.13**  
228 **unspecified**  
229 indicates that this profile does not define any constraints for the referenced CIM element
- 230 **3.14**  
231 **client**  
232 an application that exploits facilities specified by this profile



- 233 **3.15**  
234 **implementation**  
235 a set of CIM providers that realize the classes specified by this profile
- 236 **3.16**  
237 **this profile**  
238 this DMTF management profile – the *Memory Resource Virtualization Profile*
- 239 **3.17**  
240 **concrete memory resource pool**  
241 a resource pool that subdivides the capacity of its (primordial or concrete) parent resource pool
- 242 **3.18**  
243 **host memory**  
244 a contiguous extent of memory contained by the host system that may be allocated with either exclusive  
245 or shared access to a memory resource pool
- 246 **3.19**  
247 **host system**  
248 the scoping system that contains memory resources that may be allocated, virtualized, or both
- 249 **3.20**  
250 **memory composition**  
251 the aggregation of memory extents into an encompassing memory extent
- 252 **3.21**  
253 **memory resource allocation**  
254 the allocation of memory from a memory resource pool to a virtual system
- 255 **3.22**  
256 **memory resource allocation request**  
257 a request for a memory resource allocation
- 258 **3.23**  
259 **memory resource pool**  
260 a resource pool that represents memory available for memory resource allocation
- 261 **3.24**  
262 **memory resource pool configuration service**  
263 a configuration service that supports the addition or removal of host memory to or from a memory  
264 resource pool, and the creation or deletion of concrete subpools of a memory resource pool
- 265 **3.25**  
266 **primordial memory resource pool**  
267 a resource pool that aggregates memory available for or used by memory resource allocations
- 268 **3.26**  
269 **virtual computer system**  
270 the concept of a virtual system as applied to a computer system  
271 Other common industry terms are *virtual machine*, *hosted computer*, *child partition*, *logical partition*,  
272 *domain*, *guest*, or *container*.

- 273 **3.27**  
274 **virtualization platform**  
275 virtualizing infrastructure provided by a host system that enables the deployment of virtual systems
- 276 **3.28**  
277 **virtual memory**  
278 the instantiation of allocated host memory that is exposed to a virtual system through a logical memory  
279 device; the result of a memory resource allocation based on a memory resource allocation request
- 280 NOTE: The definition of the term “virtual memory” is specialized from the term “virtual resource” defined in the  
281 *Resource Allocation Profile* ([DSP1041](#)) and deviates from common computer industry parlance.

## 282 **4 Symbols and Abbreviated Terms**

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

- 284 **4.1**  
285 **CIM**  
286 Common Information Model
- 287 **4.2**  
288 **CIMOM**  
289 CIM object manager
- 290 **4.3**  
291 **MCA**  
292 capabilities settings of systems and of memory resource pools
- 293 **4.4**  
294 **MMS**  
295 mutability settings of memory resource allocation requests or memory resource allocations
- 296 **4.5**  
297 **MRA**  
298 memory resource allocation
- 299 **4.6**  
300 **MRQ**  
301 memory resource allocation request
- 302 **4.7**  
303 **RASD**  
304 CIM\_ResourceAllocationSettingData
- 305 **4.8**  
306 **VSSD**  
307 CIM\_VirtualSystemSettingData

308 **5 Synopsis**

309 **Profile Name:** Memory Resource Virtualization

310 **Version:** 1.0.0

311 **Organization:** DMTF

312 **CIM Schema Version:** 2.22

313 **Central Class:** CIM\_ResourcePool

314 **Scoping Class:** CIM\_System

315 This profile is a component profile that defines the minimum object model needed to provide for the CIM  
316 representation and management of the virtualization of memory.

317 Table 1 lists DMTF management profiles on which this profile depends.  
318

**Table 1 – Related Profiles**

Profile Name	Organization	Version	Relationship	Description
Resource Allocation	DMTF	1.1	Specializes	The abstract profile that describes the virtualization of resources See 7.2.
Allocation Capabilities	DMTF	1.0	Specializes	The abstract profile that describes capabilities for resource allocation See 7.3.
Profile Registration	DMTF	1.0	Mandatory	The profile that specifies registered profiles
System Memory	DMTF	1.0	Optional	The SMWG profile that specifies the management of system memory See 7.4.

320 **6 Description (Informative)**

321 This clause introduces the management domain addressed by the *Memory Resource Virtualization*  
322 *Profile*, and outlines the central modeling elements established for representation and control of the  
323 management domain.

324 **6.1 General**

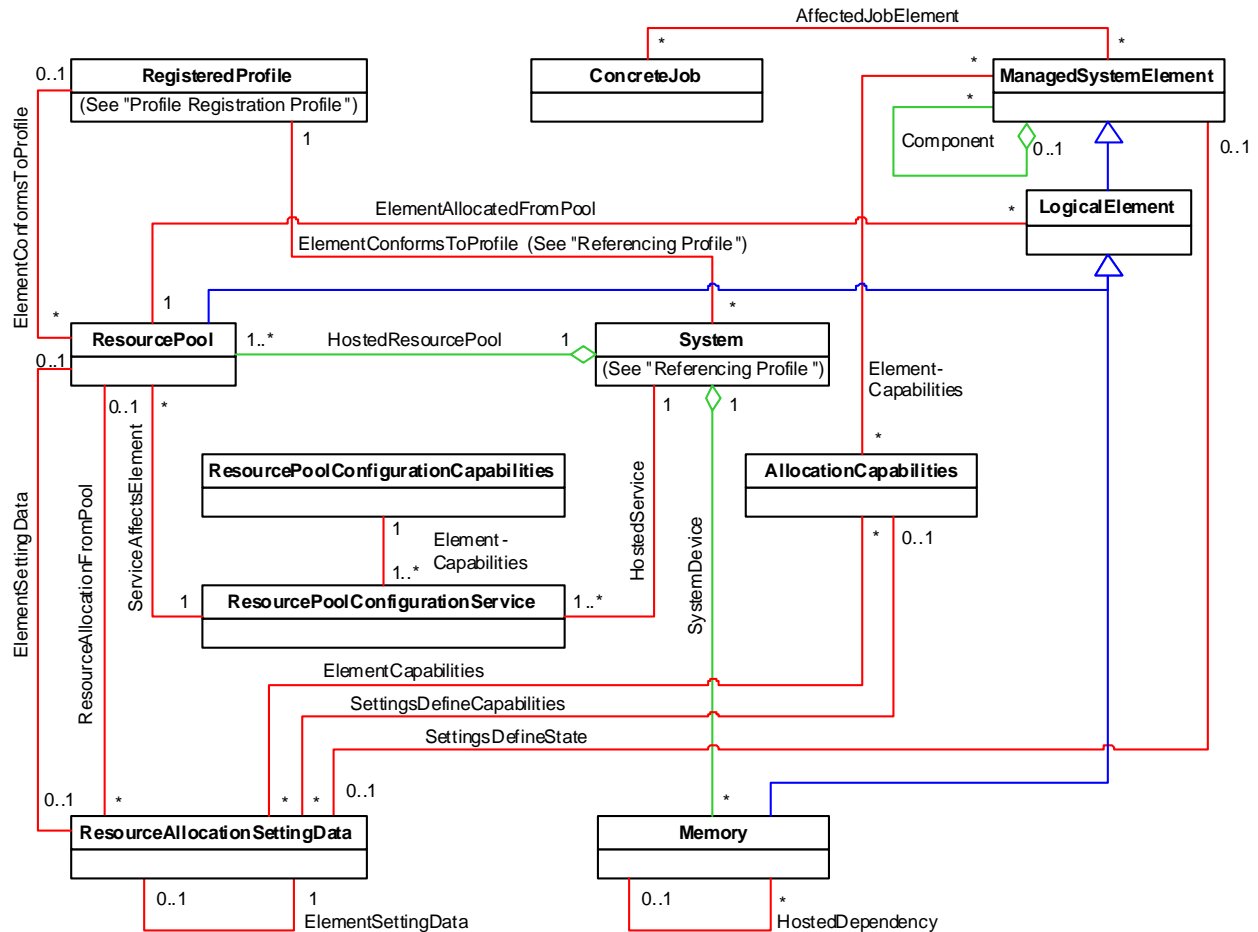
325 In computer virtualization systems, virtual computer systems are composed of component virtual  
326 resources. This profile specifies the allocation and management of host computer system memory in  
327 support of virtual computer system memory. The memory described here would appear as “physical”  
328 memory to an operating system running in the virtual computer system. This profile is not intended to  
329 specify the management of virtual memory as virtualized by operating systems.

330 This profile applies the resource virtualization pattern defined in [DSP1041](#) (*Resource Allocation Profile*)  
331 and the allocation capabilities pattern defined in [DSP1043](#) (*Allocation Capabilities Profile*) to enable the

332 management of processor resources that are allocated to virtual systems. This profile defines additional  
 333 CIM elements and constraints beyond those defined in the specialized profiles. Optionally  
 334 implementations may implement [DSP1026](#) (*System Memory Profile*) to represent host memory.

### 335 6.2 Memory Resource Virtualization Class Schema

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



341

342 **Figure 1 – Memory Resource Virtualization Profile: Profile Class Diagram**

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

- 344 • The *CIM\_ResourcePool* class modeling resource pools for memory resources. Host memory  
 345 resources are allocated from their resource pool and used to create the memory resources for  
 346 virtual systems
- 347 • The *CIM\_Component* association modeling the following relationships:
  - 348 – the relationship between memory resource pools and memory extents as components of  
 349 the resource pools

- 350 – the relationship between one aggregated memory extent and one or more aggregating  
351 memory extents
- 352 • The CIM\_ElementAllocatedFromPool association modeling hierarchies of memory resource  
353 pools and modeling the relationship of resource pools and the virtual memory allocated from  
354 those
- 355 • The CIM\_HostedResourcePool association modeling the hosting dependency between a  
356 memory resource pool and its host system. A host system supports at least one resource pool
- 357 • The CIM\_Memory class modeling the following aspects of memory:
  - 358 – memory as a device in the scope of a system, as modeled by the CIM\_SystemDevice  
359 association
  - 360 – memory as a result of a memory resource allocation from a resource pool, as modeled by  
361 the CIM\_ElementAllocatedFromPool association
  - 362 – memory as a component within memory resource pools, as modeled through the  
363 CIM\_Component association
- 364 • The CIM\_ResourceAllocationSettingData class representing memory resource allocations or  
365 memory resource allocation requests
- 366 • The CIM\_AllocationCapabilities class and the CIM\_ElementCapabilities association modeling
  - 367 – the memory resource allocation capabilities of host systems
  - 368 – the memory resource allocation capabilities of memory resource pools
  - 369 – the mutability of existing memory allocations
- 370 • The CIM\_SettingsDefineCapabilities association modeling the relation between memory  
371 allocation capabilities and the settings that define these capabilities
- 372 • The CIM\_ResourcePoolConfigurationService modeling configuration services for memory  
373 resource pools and the CIM\_ResourcePoolConfigurationCapabilities class modeling their  
374 capabilities
- 375 • The CIM\_ConcreteJob class and the CIM\_AffectedJobElement association modeling  
376 asynchronous management tasks initiated through memory resource pool configuration  
377 services

378 In general, any mention of a class in this document means the class itself or its subclasses. For example,  
379 a statement such as “an instance of the CIM\_StorageExtent class” implies an instance of the  
380 CIM\_StorageExtent class or a subclass of the CIM\_StorageExtent class.

## 381 **6.3 Memory Resource Pool**

382 This profile applies the concept of resource pools defined in [DSP1041](#) to the memory resource type. The  
383 *Memory Resource Virtualization Profile* uses the memory resource pool as the focal point for memory  
384 allocations. Virtual systems receive memory allocations from memory resource pools based on memory  
385 resource allocation requests. A memory resource pool is an aggregation of host memory available for  
386 allocation in support of virtual memory.

387 Two types of memory resource pools are defined: primordial and concrete.

### 388 **6.3.1 Primordial Memory Resource Pool**

389 A primordial memory resource pool aggregates memory capacity; it represents a subset of the  
390 manageable memory capacity of a host system.

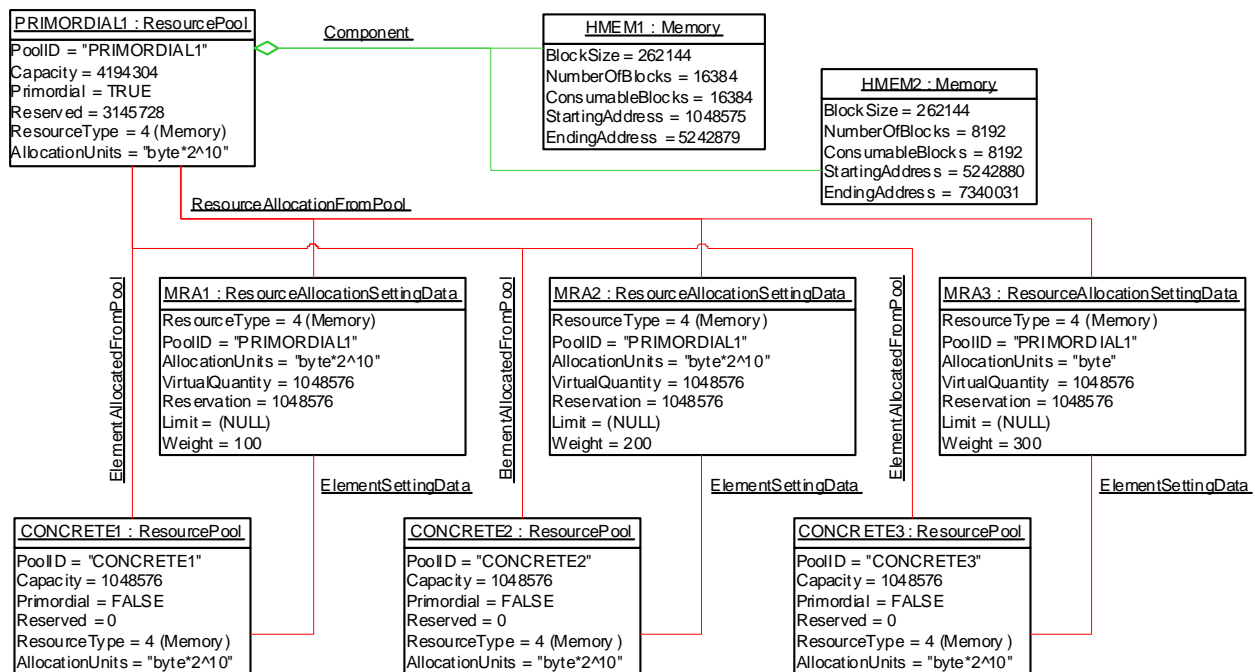
391 **6.3.2 Concrete Memory Resource Pool**

392 A concrete memory resource pool subdivides the memory capacity of its parent resource pool. The  
 393 amount of memory allocated to a concrete resource pool is less than or at most equal to the capacity of  
 394 the parent pool, but may use all of the capacity of the parent pool.

395 **6.3.3 Hierarchies of Memory Resource Pools**

396 This profile applies the concept of resource pool hierarchies defined in [DSP1041](#) to the memory resource  
 397 type; see the “Hierarchies of Resource Pools” subclause in [DSP1041](#).

398 Figure 2 shows an example of the CIM representation of a memory resource pool hierarchy in which two  
 399 host memory extents are aggregated into a primordial memory resource pool. The primordial memory  
 400 resource pool is subdivided into three concrete memory resource pools, with each concrete resource pool  
 401 allocated 1 GB of memory. The assigned weights differ for each concrete memory resource pool,  
 402 indicating different qualities of service for memory extents that are allocated from these pools.



403

404 **Figure 2 – Instance Diagram: Concept of Memory Resource Pool Hierarchies**

405 **6.3.4 Memory Resource Pool Management**

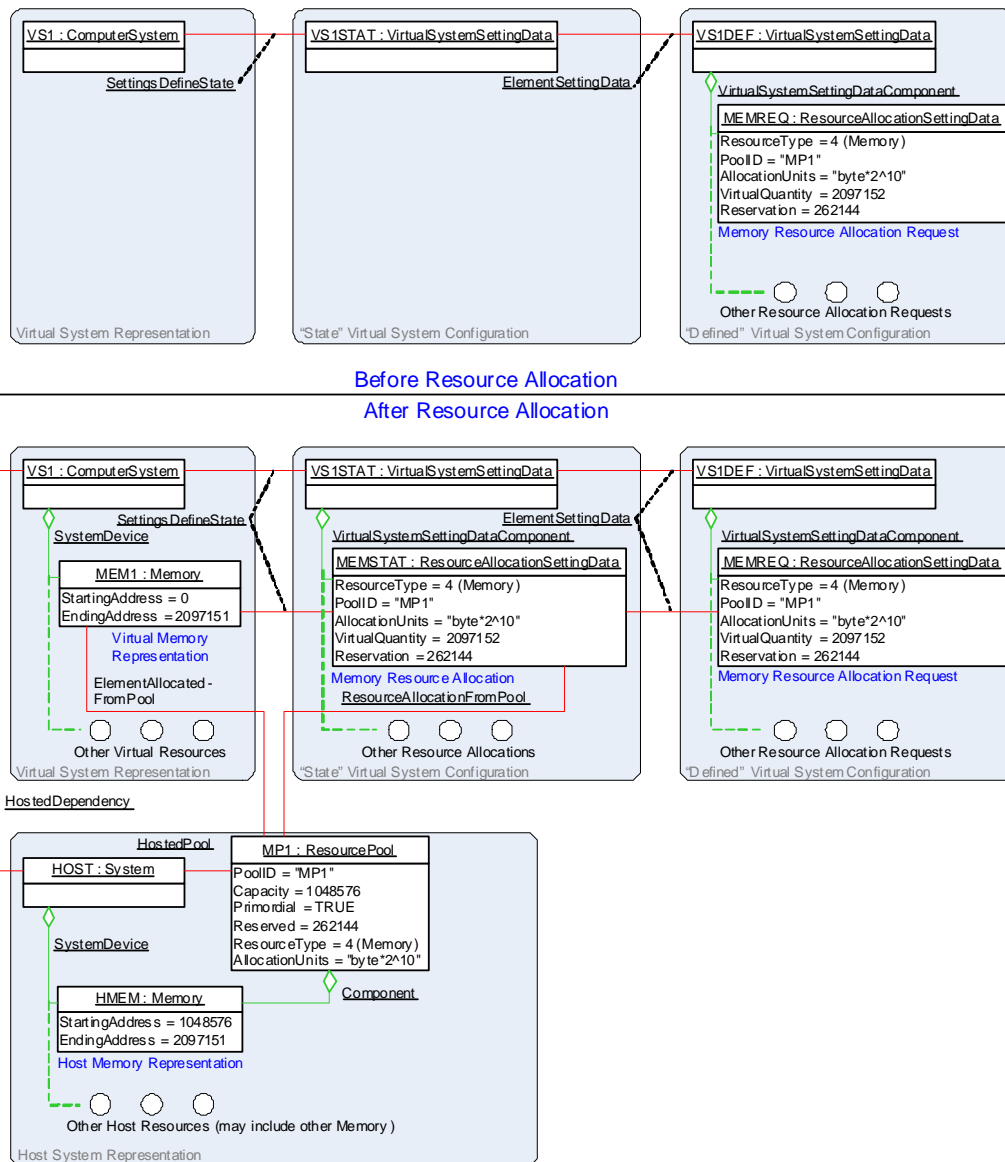
406 This profile applies the concept of resource pool management defined in [DSP1041](#) to the memory  
 407 resource type; see the “Pool and Resource Management” subclause in [DSP1041](#).

408 **6.4 Memory Resource Allocation**

409 This profile applies the concept of device resource allocation defined in [DSP1041](#) to the memory resource  
 410 type; see the “Device Resource Allocation” subclause in [DSP1041](#).

411 **6.4.1 General**

412 Figure 3 shows a typical situation for memory resource allocation in the context of a virtual system.



413

414

**Figure 3 – Instance Diagram: Concept of Memory Resource Allocation**

415 **6.4.2 Memory Resource Allocation Request**

416 The memory requirements of a virtual system are defined as part of the “Defined” virtual system  
 417 configuration. The “Defined” virtual system configuration contains memory resource allocation requests  
 418 represented as instances of the CIM\_ResourceAllocationSettingData class.

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

421 **6.4.3 Memory Resource Allocation**

422 As a virtual system is activated (or instantiated), memory needs to be allocated as requested by memory  
 423 resource allocation requests in the virtual system definition. Memory resource allocations are represented  
 424 as instances of the CIM\_ResourceAllocationSettingData class in the “State” virtual system configuration.

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

#### 427 **6.4.4 Virtual Memory**

428 Virtual memory is the instantiation of allocated host memory that is exposed to a virtual system through a  
429 logical memory device; it is the result of the memory resource allocation based on a memory resource  
430 allocation request. Virtual memory may be virtualized using techniques like paging and dynamic address  
431 translation, but may also be host memory that is directly passed through to the virtual system. Virtual  
432 memory is represented by an instance of the CIM\_Memory class as part of the virtual system  
433 representation.

434 NOTE The definition of the term “virtual memory” is specialized from the term “virtual resource” defined in [DSP1041](#)  
435 and deviates from common computer industry parlance.

436 An example of the CIM representation of virtual memory as the result of a memory resource allocation is  
437 shown on the left side in the central part of Figure 3.

#### 438 **6.4.5 Memory Virtualization**

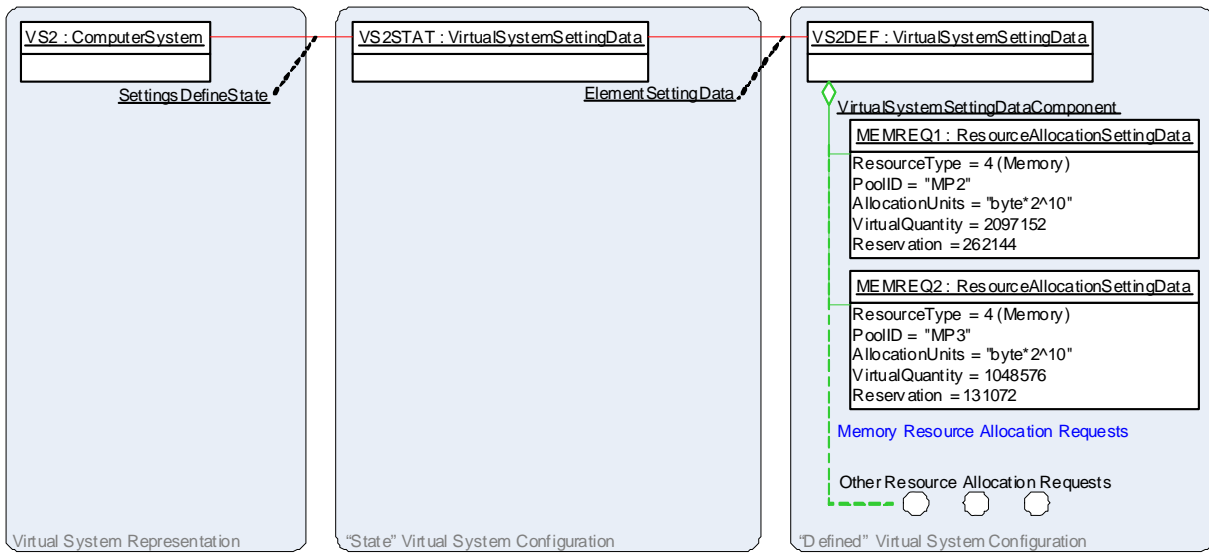
439 The amount of host memory reserved may be less than the amount of virtual memory available to the  
440 virtual system. This indicates that memory is virtualized by facilities of the host system, such that the  
441 amount of virtual memory usable by the virtual system is larger than the amount of real memory required  
442 for its support.

443 An example of the CIM representation of memory virtualization is shown in the center part of Figure 3  
444 where the amount of virtual memory is significantly larger than the amount of allocated host memory.

#### 445 **6.4.6 Memory Composition**

446 Memory may be composed of other memory. For example, a virtualization platform may support the  
447 allocation of memory from more than one resource pool, resulting in several virtual memory extents that  
448 then are composed into one aggregating memory extent. This situation is shown in Figure 4. It is similar  
449 to the situation shown in Figure 3, but in Figure 4 the virtual memory is composed from two memory  
450 extents that are allocated from two different memory resource pools. The two memory extents are  
451 composed into a memory composition that is a logical device of the virtual system.





Before Resource Allocation

After Resource Allocation

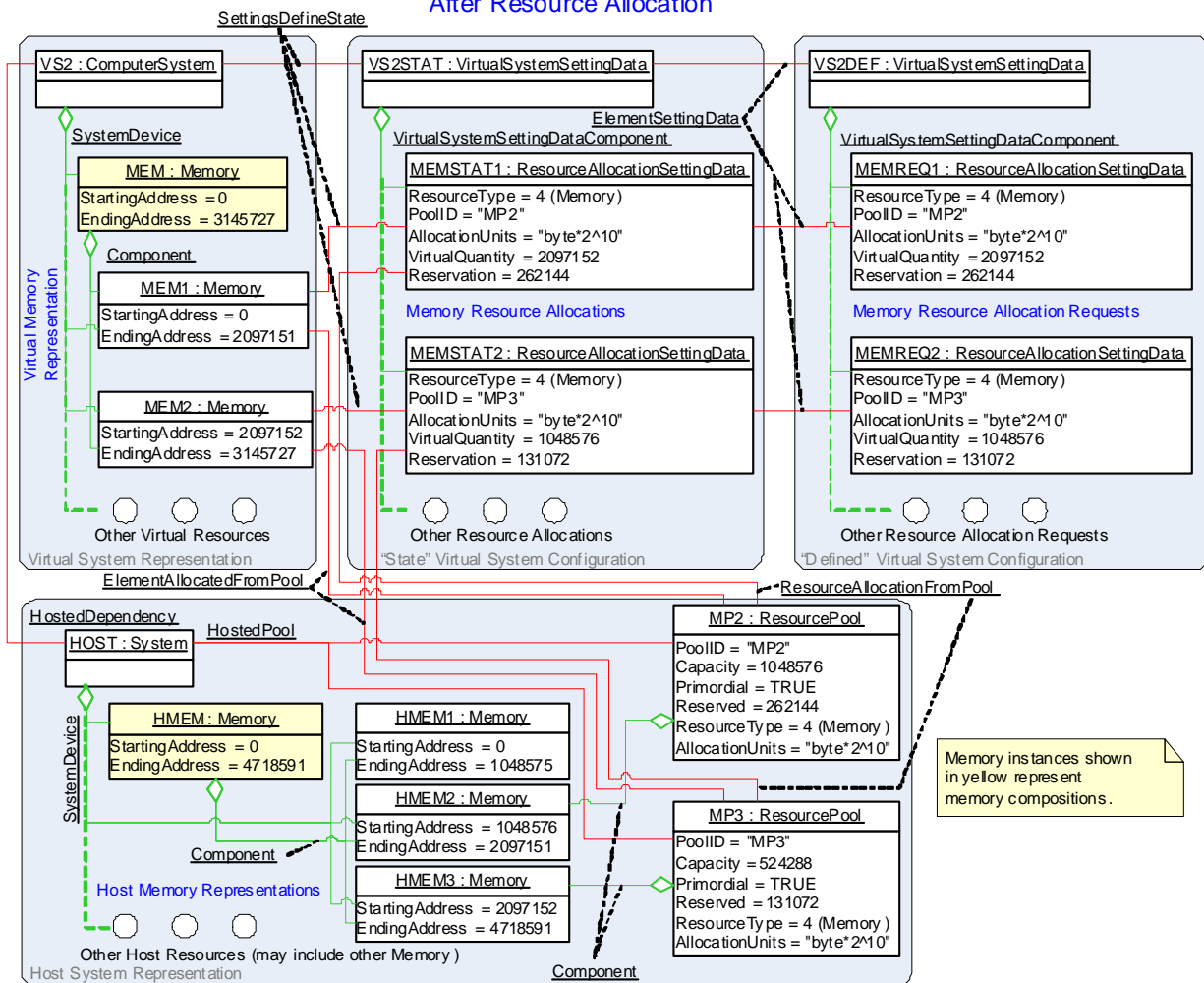


Figure 4 – Instance Diagram: Memory Composition

## 454 6.4.7 Dedicated Host Memory

455 Dedicated host memory is memory owned by the host system that is exclusively reserved for support of  
456 the virtual memory of a particular virtual system.

## 457 6.4.8 Host Memory Consumption

458 A memory resource allocation request references the memory resource pool to be used by specifying the  
459 value of the PoolID property. Host memory is allocated from the identified memory resource pool during  
460 memory resource allocation.

### 461 6.4.8.1 Statically Controlled Memory Consumption

462 With statically controlled memory consumption, a particular memory resource allocation request is  
463 supported only if the amount of host memory available in the addressed memory pool is at least as large  
464 as the amount requested. This approach is called *admission control*. Each successfully allocated memory  
465 resource reduces the amount of memory available in the pool, respectively. In the CIM representation of  
466 the memory resource pool, the amount of memory consumed from the pool is visible through the value of  
467 the Reserved property.

### 468 6.4.8.2 Dynamically Controlled Memory Consumption

469 With dynamically controlled memory consumption, the amount of host memory allocated in support of  
470 virtual memory is not a constant value but varies significantly over time, depending on factors like the  
471 memory access pattern of software executed within the virtual system or the memory consumption of  
472 other virtual machines.

473 Further, with dynamically controlled memory consumption, the amount of memory managed through a  
474 memory resource pool conceptually may be considered as unlimited, such that no admission control is  
475 performed at the time virtual memory is initially allocated (usually at virtual system activation time). Of  
476 course, the host system may experience memory shortages in later stages, such that ultimately the host  
477 system is no longer able to support the sum of virtual memory of all hosted virtual systems.

## 478 7 Implementation

479 This clause details the requirements related to classes and their properties for implementations of this  
480 profile. The CIM Schema descriptions for any referenced element and its sub-elements apply.

481 The list of all methods covered by this profile is in clause 8. The list of all classes and their elements that  
482 are covered by this profile is provided in clause 10.

483 In references to CIM Schema properties that enumerate values, the numeric value is normative and the  
484 descriptive text following it in parentheses is informative. For example, in the statement “If an instance of  
485 the CIM\_ResourcePoolConfigurationCapabilities class contains the value 4 (DeleteResourcePool is  
486 supported) in an element of the SynchronousMethodsSupported[ ] array property”, the value “4” is  
487 normative text and “(DeleteResourcePool is supported)” is informative text.

### 488 7.1 Allocation Units

489 All properties that describe storage extents of memory shall be measured in the allocation unit “kilobyte”.  
490 This requirement applies to all the following classes and properties:

- 491 • CIM\_Memory
  - 492 – StartingAddress property: Value shall be in units of kilobyte
  - 493 – EndingAddress property: Value shall be in units of kilobyte

494 NOTE: The properties NumberOfBlocks and ConsumableBlocks in the CIM\_StorageExtent class that is base class  
 495 of the CIM\_Memory class allows specifying the memory size in units of blocks, with the blocksize specified through  
 496 the value of the BlockSize property.

- 497 • CIM\_ResourceAllocationSettingData
  - 498 – AllocationUnits property: Value shall be “byte\*2^10” (equals kilobyte)
  - 499 – Reservation property: Value shall be in units of kilobyte
  - 500 – Limit property: Value shall be in units of kilobyte
- 501 • CIM\_ResourcePool
  - 502 – AllocationUnits property: Value shall be “byte\*2^10” (equals kilobyte)
  - 503 – Capacity property: Value shall be in units of kilobyte
  - 504 – Reserved property: Value shall be in units of kilobyte

## 505 7.2 Resource Allocation Profile

506 [DSP1041](#) should be used to model host memory.

507 [DSP1041](#) shall be used to model

- 508 • memory resource pool
- 509 • memory resource allocation
- 510 • memory resource allocation request
- 511 • virtual memory

### 512 7.2.1 Host Memory

513 The support of the representation of host memory is optional.

514 NOTE The support for the representation of host memory is mandatory if the *System Memory Profile* ([DSP1026](#)) is  
 515 supported; see 7.4.

516 If the representation of host memory supported, all of the following rules apply:

- 517 1) Host memory shall be represented by one or more instances of the CIM\_Memory class that are  
 518 associated with the instance of the CIM\_System class that represents the host system through  
 519 an instance of the CIM\_SystemDevice association.
- 520 2) If the host memory is composed from more than one extent, host memory shall be represented  
 521 as a memory composition as follows:
  - 522 – Each composing memory extent shall be represented by an instance of the CIM\_Memory  
 523 class as required by rule 1).
  - 524 – Total memory shall be represented by an instance of the CIM\_Memory class as required  
 525 by rule 1). In that instance the value of the StartingAddress property shall be 0, and the  
 526 value of the EndingAddress property shall be the highest ending address from any of the  
 527 composing memory extents. The range spanned by subtracting the value of the  
 528 EndingAddress property from that of the StartingAddress property shall be reflected by the  
 529 values of the BlockSize, NumberOfBlocks and ConsumableBlocks properties, respectively.
  - 530 – Each instance if the CIM\_Memory class representing a composing memory extent shall be  
 531 associated with the instance of the CIM\_Memory class representing total memory through  
 532 an instance of the CIM\_Component association.

533 NOTE: In a memory composition, total memory may span memory gaps that are not covered by a composing  
 534 memory extent.

## 535 7.2.2 Memory Resource Pool

536 This subclause specifies implementation requirements for the representation of memory resource pools.

### 537 7.2.2.1 CIM\_ResourcePool.Primordial Property

538 The value of the Primordial property shall be set to TRUE for any instance of the CIM\_ResourcePool  
539 class that represents a primordial memory resource pool. For other instances of the CIM\_ResourcePool  
540 class that represent memory resource pools, the value of the Primordial property shall be set to FALSE.

### 541 7.2.2.2 CIM\_ResourcePool.PoolID Property

542 The value of the PoolID property shall be set such that it enables unique identification of the instance of  
543 the CIM\_ResourcePool class within the scoping host system.

### 544 7.2.2.3 Aggregation of Host Resources

545 The support of the representation of the aggregation of host memory into a primordial memory resource  
546 pool is optional.

547 If the representation of the aggregation of host memory into primordial memory resource pools is  
548 supported, at least one instance of the CIM\_Component association (see 10.5) references the instance of  
549 the CIM\_ResourcePool class that represents the pool.

### 550 7.2.2.4 CIM\_ResourcePool.Reserved Property

551 The value of the Reserved property shall denote the amount of host memory that is actually reserved  
552 from the resource pool, in units of kilobyte.

### 553 7.2.2.5 CIM\_ResourcePool.Capacity Property

554 The support of the Capacity property is conditional.

555 Conditional Requirement: The Capacity property shall be supported if the aggregation of host resources  
556 is supported (see 7.2.2.3); otherwise, support of the Capacity property is optional.

557 If the Capacity property is supported, its value shall reflect the maximum amount of memory that can be  
558 allocated from the resource pool in units of kilobyte. If the instance of the CIM\_ResourcePool class  
559 represents a memory resource pool with unlimited capacity, the value of the Capacity property shall be  
560 set to the largest value supported by the uint64 datatype.

### 561 7.2.2.6 Memory Resource Pool Hierarchies

562 The support of representing memory resource pool hierarchies is optional.

563 If the representation of memory resource pool hierarchies is supported, any concrete memory resource  
564 pool shall be represented through an instance of the CIM\_ResourcePool class, where all of the following  
565 conditions shall be met:

- 566 • The value of the Primordial property shall be FALSE.
- 567 • The instance shall be associated through an instance of CIM\_ElementAllocatedFromPool  
568 association to the instance of the CIM\_ResourcePool class that represents its parent memory  
569 resource pool.
- 570 • The instance shall be associated through an instance of the CIM\_ElementSettingData  
571 association to the instance of the CIM\_ResourceAllocationSettingData class that represents the  
572 amount of memory allocated from the parent resource pool.

### 573 7.2.2.7 Default Memory Resource Pool

574 The support of designating a default memory resource pool is optional.

575 If the designation of a default memory resource pool is supported, all of the following conditions apply:

- 576 • The default memory resource pool shall be represented by an instance of the  
577 CIM\_ResourcePool class; see 10.18.
- 578 • That instance shall be associated to the instance of the CIM\_AllocationCapabilities class that  
579 represents the pools default allocation capabilities as specified in 7.3.1.5.
- 580 • The same instance of the CIM\_AllocationCapabilities class shall also represent the systems  
581 default allocation capabilities as specified in 7.3.1.2.

### 582 7.2.2.8 Memory Resource Pool Management

583 The support of memory resource pool management is optional.

#### 584 7.2.2.8.1 Indication of Support

585 If memory resource pool management is supported, the instance of the  
586 CIM\_ResourcePoolManagementCapabilities class that is associated through an instance of the  
587 CIM\_ElementCapabilities association to the instance of the CIM\_ResourcePoolManagementService that  
588 represents a memory resource pool configuration service shall represent the capabilities of the resource  
589 pool configuration service as specified in this subclause.

590 Memory resource pool management is supported (with one or more methods) if the  
591 SynchronousMethodsSupported[ ] array property, the AsynchronousMethodsSupported[ ] array property,  
592 or both have a non-NULL value and contain at least one element.

593 If memory resource pool management is not supported, the value of both the  
594 SynchronousMethodsSupported[ ] and the AsynchronousMethodsSupported[ ] array properties shall be  
595 NULL or an empty array in the instance of the CIM\_ResourcePoolManagementCapabilities class that  
596 represents the capabilities of a resource pool configuration service.

#### 597 7.2.2.8.2 ValueMap Qualifier Method Designators

598 Qualifier values defined by the ValueMap qualifiers of the SynchronousMethodsSupported[ ] and  
599 AsynchronousMethodsSupported[ ] array properties in the CIM\_ResourcePoolConfigurationCapabilities  
600 class shall designate methods as follows:

- 601 • The value 3 (CreateChildResourcePool is supported) designates the  
602 CreateChildResourcePool( ) method.
- 603 • The value 4 (DeleteResourcePool is supported) designates the DeleteResourcePool( ) method.
- 604 • The value 5 (AddResourcesToResourcePool is supported) designates the  
605 AddResourcesToResourcePool( ) method.
- 606 • The value 6 (RemoveResourcesFromResourcePool) designates the  
607 RemoveResourcesFromResourcePool( ) method.

#### 608 7.2.2.8.3 Implementation Requirements

609 Elements in the value sets of the SynchronousMethodsSupported[ ] and  
610 AsynchronousMethodsSupported[ ] array properties in the CIM\_ResourcePoolConfigurationCapabilities  
611 class shall be specified according to the following rules:

- 612 • A particular ValueMap qualifier value shall appear as an element in the value set of at most one  
613 array property.

- 614       • If a particular ValueMap qualifier value does not appear in the value set of either array property,  
615       the corresponding method is not supported.
- 616       • If a particular qualifier value is used as element in the value set of the  
617       SynchronousMethodsSupported[ ] array property, the corresponding method shall be supported  
618       with synchronous behavior only.
- 619       • If a particular qualifier value is used as element in the value set of the  
620       AsynchronousMethodsSupported[ ] array property, the corresponding method shall be  
621       supported with synchronous behavior, asynchronous behavior, or both.

622 A method implementation shall apply following rules:

- 623       • A supported method (with synchronous or asynchronous behavior) shall not return 1 (Not  
624       Supported).
- 625       • A method supported with synchronous behavior only shall not return 4096 (Method Parameters  
626       Checked – Job Started).

#### 627 **7.2.2.8.4 Availability of Support of Asynchronous Operations**

628 This subclause specifies the CIM elements that indicate the availability of the support for asynchronous  
629 operations for memory resource pool management.

630 Asynchronous operations for memory resource pool management are supported (with one or more  
631 methods) if the AsynchronousMethodsSupported[ ] array property has a non-NULL value and contains at  
632 least one element.

### 633 **7.2.3 Memory Resource Allocation**

634 NOTE: [DSP1041](#) specifies two alternatives for modeling resource allocation: *simple resource allocation* and *virtual*  
635 *resource allocation*.

636 Implementations conforming to the *Memory Resource Virtualization Profile* shall implement virtual  
637 resource allocation as defined in the “Modeling Virtual Resource Allocation (Optional)” clause of  
638 [DSP1041](#).

### 639 **7.2.4 CIM\_ResourceAllocationSettingData**

640 This subclause specifies the use of the CIM\_ResourceAllocationSettingData class.

#### 641 **7.2.4.1 General**

642 Instances of the CIM\_ResourceAllocationSettingData class

- 643       • shall be used to represent memory resource allocation requests (MRQ)
- 644       • shall be used to represent memory resource allocations (MRA)
- 645       • shall be used to represent settings that define the capabilities of systems and of memory  
646       resource pools (MCA)
- 647       • may be used to represent settings that define the mutability of memory resource allocations  
648       (MMS)

649 The specifications in this subclause generally define constraints for property values used in these  
650 representations. Constraints that apply to only a subset of these representations are prefixed with the  
651 respective acronyms and followed by the word “Only” and a colon.

#### 652 7.2.4.2 **CIM\_ResourceAllocationSettingData.PoolID Property**

653 The value of the PoolID property shall designate the memory resource pool. A NULL value shall indicate  
654 the use of the host system's default memory resource pool.

#### 655 7.2.4.3 **CIM\_ResourceAllocationSettingData.ConsumerVisibility Property**

656 The value of the ConsumerVisibility property shall denote whether host memory is directly passed  
657 through to the virtual system or whether memory is virtualized. Values shall be assigned as follows:

- 658 • A value of 2 (Passed-Through) shall denote that host memory is passed through.
- 659 • A value of 3 (Virtualized) shall denote that memory is virtualized.
- 660 • MRQ Only: A value of 0 (Unknown) shall be used if the represented memory resource allocation  
661 request does not predefine which type of memory shall be allocated.

662 Other values shall not be used.

#### 663 7.2.4.4 **CIM\_ResourceAllocationSettingData.HostResource[ ] Array Property**

664 Support of the HostResource[ ] array property is conditional if representing memory resource allocations,  
665 and optional otherwise.

666 MRA Only: Conditional Requirement: The HostResource[ ] array property shall be supported if the value  
667 of the MappingBehavior property is 2 (Dedicated).

668 Otherwise, the HostResource[ ] array property may be supported.

669 If HostResource[ ] array property is supported, the following rules apply:

- 670 • If the value of the ConsumerVisibility property (see 7.2.4.3) is 2 (Passed-Through), the value of  
671 the HostResource[ ] array property should designate the host memory resource that is passed  
672 through to a virtual system. A value of NULL or an empty array should indicate that the passed-  
673 through host memory resource is not represented by an instance of the CIM\_Memory class.
- 674 • If the value of the ConsumerVisibility property is 3 (Virtualized), the value of the HostResource[ ]  
675 array property shall be NULL or shall be an empty array.
- 676 • MRA Only: The value of the HostResource[ ] array property depends on the value of the  
677 MappingBehavior property as follows:
  - 678 – If the value of the MappingBehavior property is 2 (Dedicated), elements in the value of the  
679 HostResource[ ] array property shall reference instances of the CIM\_Memory class that  
680 represent host memory extents that are exclusively dedicated to the virtual system.
  - 681 – If the value of the MappingBehavior property is 3 (Hard Affinity) or 4 (Soft Affinity),  
682 elements in the value of the HostResource[ ] array property shall reference instances of the  
683 CIM\_Memory class that represent host memory extents that provide the allocation of the  
684 virtual memory.
- 685 • MRQ Only: The value of the HostResource[ ] array property depends on the value of the  
686 MappingBehavior property as follows:
  - 687 – If the value of the MappingBehavior property is 2 (Dedicated), elements in the value of the  
688 HostResource[ ] array property shall reference instances of the CIM\_Memory class that  
689 represent host memory extents that are required with dedicated access by the virtual  
690 system.
  - 691 – If the value of the MappingBehavior property is 3 (Soft Affinity), elements in the value of the  
692 HostResource[ ] array property shall reference instances of the CIM\_Memory class that

693 represent host memory extents that are preferred for the allocation of the virtual systems  
694 memory.

695 – If the value of the MappingBehavior property is 4 (Hard Affinity), elements in the value of  
696 the HostResource[ ] array property shall reference instances of the CIM\_Memory class that  
697 represent host memory extents that are required for the allocation of the virtual systems  
698 memory.

699 If the HostResource[ ] array property is not supported, the value of the HostResource[ ] array property  
700 shall be NULL. This indicates that host memory extents that are exclusively dedicated to the virtual  
701 system or that provide the allocation of the virtual memory are not defined.

#### 702 **7.2.4.5 CIM\_ResourceAllocationSettingData.VirtualQuantity Property**

703 The value of the VirtualQuantity property shall denote the amount of virtual memory available to a virtual  
704 system in units of kilobyte.

#### 705 **7.2.4.6 CIM\_ResourceAllocationSettingData.Reservation Property**

706 Support of the Reservation property is optional.

707 If the Reservation property is supported, the value of the Reservation property shall denote the amount of  
708 host memory reserved for the exclusive use of a virtual system in units of kilobyte.

709 If the Reservation property is not supported, it shall have a value of NULL. This indicates that an amount  
710 of host memory reserved for the exclusive use of the virtual system is not defined.

#### 711 **7.2.4.7 CIM\_ResourceAllocationSettingData.Limit Property**

712 Support of the Limit property is optional.

713 If the Limit property is supported, the value of the Limit property shall denote the maximum amount of  
714 host memory available to a virtual system in units of kilobyte.

715 If the Limit property is not supported, it shall have a value of NULL. This indicates that a maximum  
716 amount of host memory available to the virtual system is not defined.

#### 717 **7.2.4.8 CIM\_ResourceAllocationSettingData.Weight Property**

718 Support of the Weight property is optional.

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

721 If the Weight property is not supported, it shall have a value of NULL. This indicates that a relative priority  
722 of the memory resource allocation in relations to other memory resource allocations is not defined.

#### 723 **7.2.4.9 CIM\_ResourceAllocationSettingData.Parent Property**

724 The implementation of the Parent property is optional.

725 If the Parent property is supported, the value of the Parent property shall denote the parent entity of the  
726 memory resource allocation.

727 If the Parent property is not supported, is shall have a value of NULL. This indicates that a parent entity of  
728 the memory resource allocation is not defined.

729 NOTE: For example, the value of the Parent property may refer to the name of an address space that resides in  
730 the host system.



#### 731 **7.2.4.10 CIM\_ResourceAllocationSettingData.Connection[ ] Array Property**

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

733 If Connection[ ] array property is supported, its value shall contain elements that identify entities  
734 connected to the memory resource allocation.

735 If the Connection[ ] array property is not supported, it shall have a value of NULL. This indicates that  
736 entities connected to the memory resource allocation are not defined.

737 NOTE: For example, elements of the value of the Connection[ ] array property may refer to the name of shared  
738 memory segments that are mapped to the allocated virtual memory.

#### 739 **7.2.4.11 CIM\_ResourceAllocationSettingData.MappingBehavior Property**

740 The implementation of the MappingBehavior property is optional.

741 If the MappingBehavior property is supported, its value shall denote how host resources referenced by  
742 elements in the value of HostResource[ ] array property relate to the memory resource allocation. The  
743 following rules apply:

- 744 • MRA Only:
  - 745 – A value of 2 (Dedicated) shall indicate that the represented memory resource allocation is
  - 746 provided by host memory resources as referenced by the value of the HostResource[ ]
  - 747 array property that are exclusively dedicated to the virtual system.
  - 748 – A value of 3 (Soft Affinity) or 4 (Hard Affinity) shall indicate that the represented memory
  - 749 resource allocation is provided using host memory resource as referenced by the value of
  - 750 the HostResource[ ] array property.
  - 751 – Other values shall not be used.
- 752 • MRQ Only:
  - 753 – A value of 0 (Unknown) shall indicate that the memory resource allocation request does
  - 754 not require specific host resources.
  - 755 – A value of 2 (Dedicated) shall indicate that the memory resource allocation request shall be
  - 756 provided by exclusively dedicated host memory resources as specified through the value of
  - 757 the HostResource[ ] array property.
  - 758 – A value of 3 (Soft Affinity) shall indicate that the memory resource allocation request shall
  - 759 preferably be provided by host memory resources as specified through the value of the
  - 760 HostResource[ ] array property, but that other resources may be used if the requested
  - 761 resources are not available.
  - 762 – A value of 4 (Hard Affinity) shall indicate that the memory resource allocation request shall
  - 763 preferably be provided by host memory resources as specified through the value of the
  - 764 HostResource[ ] array property and that other resources shall not be used if the requested
  - 765 resources are not available.
  - 766 – Other values shall not be used.

767 If the MappingBehavior property is not supported, it shall have a value of NULL. This indicates that a  
768 further qualification of the value of the HostResource[ ] array property through the value of the  
769 MappingBehavior property is not defined.

### 770 **7.2.5 Virtual Memory**

771 For the representation of virtual memory all of the following rules apply:

- 772 1) Virtual memory shall be represented by one or more instances of the CIM\_Memory class that  
773 are associated with the instance of the CIM\_ComputerSystem class that represents the virtual  
774 system through an instance of the CIM\_SystemDevice association.
- 775 2) If the virtual memory is composed from more than one extent, virtual memory shall be  
776 represented as a memory composition as follows:
- 777 – Each composing memory extent shall be represented by an instance of the CIM\_Memory  
778 class as required by rule 1).
  - 779 – Total memory shall be represented by an instance of the CIM\_Memory class as required  
780 by rule 1). In that instance the value of the StartingAddress property shall be 0, and the  
781 value of the EndingAddress property shall be the highest ending address from any of the  
782 composing memory extents. The range defined by subtracting the value of the  
783 EndingAddress property from that of the StartingAddress property shall be reflected by the  
784 values of the BlockSize, NumberOfBlocks and ConsumableBlocks properties, respectively.
  - 785 – Each instance of the CIM\_Memory class representing a composing memory extent shall be  
786 associated with the instance of the CIM\_Memory class representing total memory through  
787 an instance of the CIM\_Component association.
- 788 3) If a memory extent is directly based on a memory resource allocation, the instance of the  
789 CIM\_Memory class representing that extent shall be associated to all of the following instances:
- 790 – the instance of the CIM\_ResourceAllocationSettingData that represents memory resource  
791 allocation through an instance of the CIM\_SettingsDefineState association
  - 792 – the instance of the CIM\_ResourcePool that represents the memory resource pool providing  
793 the resource allocation through an instance of the CIM\_ElementAllocatedFromPool  
794 association

795 NOTE 1: In a memory composition, total memory may span memory gaps that are not covered by a composing  
796 memory extent. Discontiguous memory as seen in the memory address space presented to the virtual system is not  
797 supported.

798 NOTE 2: In a memory composition, total memory is never the direct result of a memory allocation; instead, each  
799 composing memory extent is the direct result of a memory allocation.

800 Additional constraints apply if [DSP1026](#) is implemented; see 7.4.

## 801 7.3 Allocation Capabilities Profile

802 [DSP1043](#) shall be used to model the following aspects:

- 803 • the memory resource allocation capabilities of host systems
- 804 • the memory resource allocation capabilities of memory resource pools
- 805 • the mutability of memory resource allocations or memory resource allocation requests

### 806 7.3.1 Memory Resource Allocation Capabilities

807 The memory resource allocation capabilities of host systems and of memory resource pools shall be  
808 represented by instances of the CIM\_AllocationCapabilities class.

#### 809 7.3.1.1 CIM\_AllocationCapabilities Class (Capabilities)

810 This subclause specifies the use of the CIM\_AllocationCapabilities class for the representation of the  
811 memory resource allocation capabilities of host systems and of memory resource pools.

#### 812 **7.3.1.1.1 Relationship of System and Resource Pool Capabilities**

813 The memory allocation capabilities of a host system shall be a superset of the memory allocation  
814 capabilities of all memory resource pools that are hosted by the host system.

#### 815 **7.3.1.1.2 CIM\_AllocationCapabilities.RequestedTypesSupported Property**

816 The value of the RequestedTypesSupported property shall indicate whether the host system or the  
817 resource pool support memory resource allocation requests for particular host resources, as follows:

- 818 • A value of 2 (Specific) shall indicate support of specific requests that only refer to a specific host  
819 memory resource.
- 820 • A value of 3 (General) shall indicate support of generic memory requests that do not refer to a  
821 particular host memory resource.
- 822 • A value of 4 (Both) shall indicate support of both specific and generic memory requests.

823 Other values shall not be used.

#### 824 **7.3.1.1.3 CIM\_AllocationCapabilities.SharingMode Property**

825 The value of the SharingMode property shall indicate whether the host system or the resource pool  
826 support exclusive access or shared use of managed memory resources, as follows:

- 827 • A value of 2 (Dedicated) shall indicate support of memory resources with exclusive access. This  
828 value shall be used if host memory is allocated directly to a virtual system for exclusive use.
- 829 • A value of 3 (Shared) shall indicate support of memory resources with shared access. This  
830 value shall be used if host memory is not allocated directly to a virtual system.

831 Other values shall not be used.

832 NOTE: More than one instance of the CIM\_AllocationCapabilities class may be associated with a host system or  
833 resource pool. As a result, support of both shared and exclusive access to memory resources may be modeled, and  
834 one of these sharing modes may be designated the default sharing mode; see 7.3.1.3.

#### 835 **7.3.1.2 Host System Memory Resource Allocation Capabilities**

836 Each instance of the CIM\_System class that represents a host system shall be associated through the  
837 CIM\_ElementCapabilities association with one or more instances of the CIM\_AllocationCapabilities class  
838 that represent the host system's memory resource allocation capabilities. One of these instances shall  
839 represent the default memory resource allocation capabilities as specified in 7.3.1.3.

#### 840 **7.3.1.3 Default Host System Memory Resource Allocation Capabilities**

841 Exactly one instance of the CIM\_AllocationCapabilities class shall exist that describes the default memory  
842 resource allocation capabilities of a host system. That instance shall be associated with the instance of  
843 the CIM\_System class that represents the host system through an instance of the  
844 CIM\_ElementCapabilities association where the value of the Characteristics[ ] array property shall contain  
845 exactly one element, and that element shall have a value of 2 (Default).

#### 846 **7.3.1.4 Memory Resource Pool Memory Resource Allocation Capabilities**

847 Each instance of the CIM\_ResourcePool class that represents a memory resource pool shall be  
848 associated through the CIM\_ElementCapabilities association with one or more instances of the  
849 CIM\_AllocationCapabilities class that represent the memory resource pool's memory resource allocation  
850 capabilities. One of these instances shall represent the default memory resource allocation capabilities as  
851 specified in 7.3.1.5.

### 852 7.3.1.5 Default Memory Resource Pool Memory Resource Allocation Capabilities

853 Exactly one instance of the CIM\_AllocationCapabilities class shall exist that describes the default memory  
854 resource allocation capabilities of a memory resource pool. That instance shall be associated with the  
855 instance of the CIM\_ResourcePool class that represents the memory resource pool through an instance  
856 of the CIM\_ElementCapabilities where the value of the Characteristics array property shall contain exactly  
857 one element, and that element shall have a value of 2 (Default).

## 858 7.3.2 Memory Resource Allocation Mutability

859 The support for the representation of the mutability of memory resource allocation requests and memory  
860 resource allocations is optional.

### 861 7.3.2.1 Indication of Support

862 If the representation of the mutability of memory resource allocation requests and memory resource  
863 allocations is supported, in both cases it shall be represented by instances of the  
864 CIM\_AllocationCapabilities class that are associated with the instance of the  
865 CIM\_ResourceAllocationSettingData class that represents the memory resource allocation request or  
866 memory resource allocation through instances of the CIM\_ElementCapabilities association.

867 If the representation of the mutability of a memory resource allocation or a memory resource allocation  
868 request is not supported, the instance of the CIM\_ResourceAllocationSettingData class that represents  
869 the memory resource allocation request or the memory resource allocation shall not be associated with  
870 an instance of the CIM\_AllocationCapabilities class through an instance of the CIM\_ElementCapabilities  
871 association.

### 872 7.3.2.2 CIM\_AllocationCapabilities Class (Mutability)

873 This subclause specifies the use of the CIM\_AllocationCapabilities class for the representation of the  
874 mutability of a memory resource allocation request or a memory resource allocation.

#### 875 7.3.2.2.1 CIM\_AllocationCapabilities.RequestedTypesSupported Property

876 The value of the RequestedTypesSupported property shall indicate whether the type of the memory  
877 resource allocation can be changed, as follows:

- 878 • A value of 2 (Specific) shall indicate support of change requests that refer to a specific host  
879 memory resource.
- 880 • A value of 3 (General) shall indicate support of change requests that do not refer to a particular  
881 host memory resource.
- 882 • A value of 4 (Both) shall indicate support of either type of change request.

883 Other values shall not be used.

#### 884 7.3.2.2.2 CIM\_AllocationCapabilities.SharingMode Property

885 The value of the SharingMode property shall indicate whether the sharing mode of the memory resource  
886 allocation can be changed, as follows:

- 887 • A value of NULL shall indicate that the sharing mode of the memory allocation can not be  
888 changed.
- 889 • A value of 2 (Dedicated) shall indicate support of requests to change the sharing mode to  
890 exclusive access.
- 891 • A value of 3 (Shared) shall indicate support of requests to change the sharing mode to shared  
892 access.

893 Other values shall not be used.

#### 894 **7.3.2.2.3 CIM\_AllocationCapabilities.SupportedAddStates[ ] Array Property**

895 If the addition of memory to the described memory resource allocation or memory resource allocation  
896 request is not supported, then If the value of the SupportedAddStates[ ] array property shall be NULL.

897 If the value set of the SupportedAddStates[ ] is empty, then the addition of memory to the described  
898 memory resource allocation or memory resource allocation request shall be supported regardless of the  
899 virtual system state of the scoping virtual system.

900 If values are provided as elements of the SupportedAddStates[ ] array property, these values shall  
901 designate a set of potential virtual system states. The addition of memory to the described memory  
902 resource allocation or memory resource allocation request shall be supported if the scoping virtual system  
903 is in any of the designated virtual system states.

#### 904 **7.3.2.2.4 CIM\_AllocationCapabilities.SupportedRemoveStates[ ] Array Property**

905 If the removal of memory from the described memory resource allocation or memory resource allocation  
906 request is not supported, then the value of the SupportedRemoveStates[ ] array property shall be NULL.

907 If the value set of the SupportedRemoveStates[ ] array property is empty, then the removal of memory  
908 from the described memory resource allocation or memory resource allocation request shall be supported  
909 regardless of the virtual system state of the scoping virtual system.

910 If values are provided as elements of the SupportedRemoveStates[ ] array property, these values shall  
911 designate a set of potential virtual system states. The removal of memory from the described memory  
912 resource allocation or memory resource allocation request shall be supported if the scoping virtual system  
913 is in any of the designated virtual system states.

### 914 **7.4 System Memory Profile**

915 The support of [DSP1026](#) for representation of host system memory is optional.

916 The support of [DSP1026](#) for representation of virtual system memory is optional. Additional constraints  
917 defined in 7.2.5 apply.

918 NOTE: [DSP1026](#) defines that there is exactly one instance of the CIM\_Memory class associated to the instance of  
919 the CIM\_System class representing a system through an instance of the CIM\_SystemDevice association. In context  
920 of the *Memory Resource Virtualization Profile* it is expected that there are host systems as well as virtual systems  
921 that have more than one memory extent assigned. A possible solution would be to use a memory composition where  
922 the instance of the CIM\_Memory class that represents the memory composition is assigned as central instance of  
923 [DSP1026](#).

## 924 **8 Methods**

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

### 927 **8.1 General**

928 Support of intrinsic and extrinsic methods is specified by the “Methods” clauses in [DSP1041](#) and  
929 [DSP1043](#). The only class added through the *Memory Resource Virtualization Profile* is the  
930 CIM\_RegisteredProfile class; see 10.15.

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

## 933 8.2 Profile conventions for operations

934 For each profile class (including associations), the implementation requirements for operations, including  
935 for those in the following default list, are specified in class-specific subclauses of this clause.

936 The default list of operations for all classes is:

937     GetInstance( )

938     EnumerateInstances( )

939     EnumerateInstanceNames( )

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

941     Associators( )

942     AssociatorNames( )

943     References( )

944     ReferenceNames( )

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

## 950 8.3 CIM\_RegisteredProfile

951 All operations in the default list in 8.2 are supported as described by [DMTF DSP0200](#).

# 952 9 Use Cases (Informative)

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

## 955 9.1 Object Diagram

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

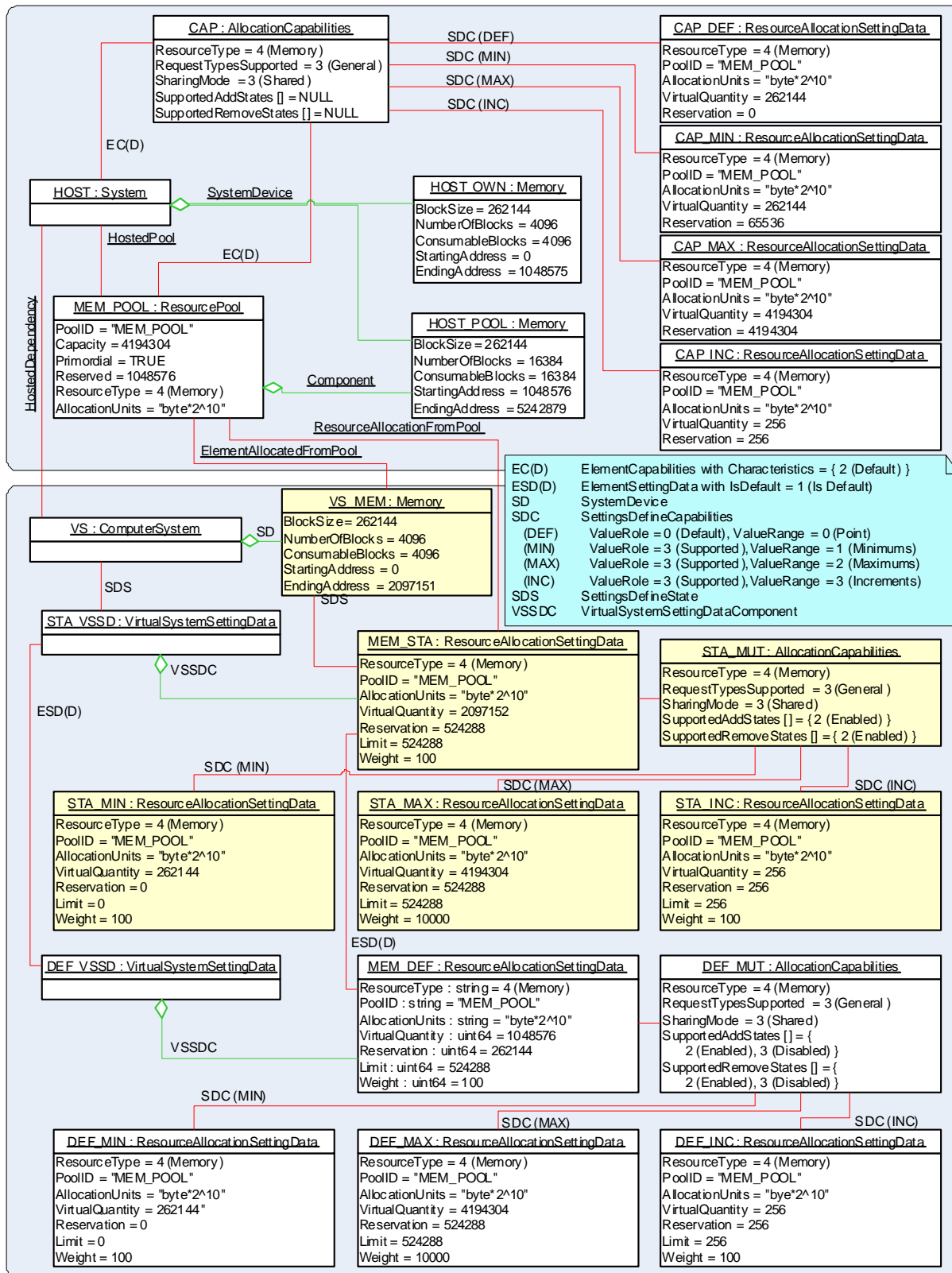
958 In Figure 5 the host system is represented by an instance of the CIM\_System class tagged HOST. The  
959 host system owns two memory resources represented by instances of the CIM\_Memory class that are  
960 tagged HOST\_OWN and HOST\_POOL. The first instance represents a memory extent in the lower part  
961 of the host memory that is used exclusively by the host because it is not assigned to a memory resource  
962 pool. The second instance represents a memory extent in the higher part of the host memory that is  
963 assigned to a memory resource pool.

964 The host system hosts a primordial memory resource pool represented by an instance of the  
965 CIM\_ResourcePool class tagged MEM\_POOL; in that instance the value of the PoolID property is  
966 "MEM\_POOL". The host memory resource that is represented by the instance of the CIM\_Memory class  
967 tagged HOST\_POOL is aggregated into the pool through an instance of the CIM\_Component association.

968 The allocation capabilities of the host system and of the memory pool are represented by the same  
969 instance of the CIM\_AllocationCapabilities class tagged CAP. Four instances of the  
970 CIM\_ResourceAllocationSettingData class (tagged CAP\_DEF, CAP\_MIN, CAP\_MAX, and CAP\_INC) are  
971 associated with that instance through instances of the CIM\_SettingsDefineCapabilities association. The

972 values of the ValueRange and ValueRole properties in the association instances designate the  
973 referenced instances of the CIM\_ResourceAllocationSettingData class that represent the default,  
974 minimum, maximum, and increment for memory resource allocations that are supported by the system  
975 and the pool.

976 The host system hosts a virtual system that is represented by an instance of the CIM\_ComputerSystem  
977 class tagged VS. The hosted relationship is shown through an instance of the CIM\_HostedDependency  
978 association.



979

980 Figure 5 – Instance Diagram: Example CIM Representation of Memory Resource Virtualization



981 The head element of the “State” virtual system configuration is the instance of the  
 982 CIM\_VirtualSystemSettingData class tagged STA\_VSSD; it is associated with the instance of the  
 983 CIM\_ComputerSystem class through an instance of the CIM\_SettingsDefineState association. The  
 984 “State” virtual system configuration contains an element of the CIM\_ResourceAllocationSettingData class  
 985 tagged MEM\_STA that represents the memory resource allocation assigned to the virtual system. The  
 986 virtual memory that results from the memory resource allocation is represented as part of the virtual  
 987 system representation by the instance of the CIM\_Memory class tagged VS\_MEM.

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

991 The head element of the “Defined” virtual system configuration is the instance of the  
 992 CIM\_VirtualSystemSettingData class tagged DEF\_VSSD; it is associated with the head element of the  
 993 “State” virtual system configuration through an instance of the CIM\_ElementSettingData association  
 994 where the value of the IsDefault property is 1 (Is Default). The “Defined” virtual system configuration  
 995 contains an element of the CIM\_ResourceAllocationSettingData class tagged MEM\_DEF that represents  
 996 the memory resource allocation request that defines the memory requirements of the virtual system. That  
 997 definition is used when the virtual system is activated and host memory is allocated to support the virtual  
 998 system’s virtual memory.

999 The example in Figure 5 shows a situation in which the VirtualQuantity property in the instances of the  
 1000 CIM\_ResourceAllocationSettingData class in the “State” and the “Defined” virtual system configuration  
 1001 has different values. This indicates that the memory size was dynamically modified (for example, by an  
 1002 operator command) sometime after the virtual system activation. This reflects a temporary situation that is  
 1003 retained until the virtual system is recycled, at which point memory resources are deallocated and then  
 1004 newly allocated based on the memory resource allocation request in the virtual system definition.

1005 The mutability of both the memory resource allocation request in the “Defined” virtual system  
 1006 configuration and of the memory resource allocation in the “State” virtual system configuration is  
 1007 represented by instances of the CIM\_ResourceAllocationSettingData class associated through  
 1008 respectively parameterized instances of the CIM\_SettingsDefineCapabilities association.

1009 Acceptable virtual system states for the addition and the removal of virtual memory are different for the  
 1010 memory resource allocation request and the memory resource allocation. The memory resource  
 1011 allocation can be modified only while the virtual system remains instantiated, as indicated by a value of  
 1012 2 (Enabled) in the instance of the CIM\_AllocationCapabilities class tagged STA\_MUT.

## 1013 9.2 Inspection

1014 This set of use cases describes how to obtain various CIM instances that represent memory-related  
 1015 information of host and virtual systems.

### 1016 9.2.1 Obtain the Memory Size of an Active Virtual System

1017 **Assumption:** All of the following:

- 1018 • The client knows a reference to the instance of the CIM\_ComputerSystem class that represents  
 1019 the virtual system.
- 1020 • The virtual system is in a virtual system state other than “Defined” (that is, is instantiated), which  
 1021 is indicated by a value other than 3 (Disabled) for the EnabledState property in the instance of  
 1022 the CIM\_ComputerSystem class.

1023 The sequence of activities is as follows:

- 1024 1) The client resolves the CIM\_SystemDevice association to find instances of the CIM\_Memory  
 1025 class that represent the virtual memory, invoking the intrinsic Associators( ) CIM operation with  
 1026 parameter values set as follows:

- 1027           – The value of the ObjectName parameter refers to the instance of the  
1028           CIM\_ComputerSystem class that represents the virtual system.
- 1029           – The value of the AssocClass parameter is set to “CIM\_SystemDevice”.
- 1030           – The value of the ResultClass parameter is set to “CIM\_Memory”.
- 1031           The result of step 1) is a set of instances of the CIM\_Memory class.
- 1032           2) For each of the instances returned from step 1), the client inspects the values of the BlockSize  
1033           and NumberOfBlocks properties, multiplies these values, and adds the results. The resulting  
1034           sum is the amount of virtual memory available to the virtual system.

1035   **Result:** The client knows the amount of virtual memory available to the virtual system.

1036   In the example CIM representation shown in Figure 5, the client initially would know the instance of the  
1037   CIM\_ComputerSystem class tagged VS that represents the virtual system. From there, the client would  
1038   follow the CIM\_SystemDevice association to the instance of the CIM\_Memory class tagged VS\_MEM that  
1039   represents the only virtual memory resource allocated to the virtual system. The client would multiply the  
1040   value of the BlockSize property (524288) with the value of the ConsumableBlocks property (4096),  
1041   yielding a result of 2147483648 bytes or 2097152 KB for the virtual system memory size.

## 1042   **9.2.2 Obtain the Memory Size of a Defined Virtual System**

1043   **Assumption:** All of the following:

- 1044           • The client knows a reference to the instance of the CIM\_ComputerSystem class that represents  
1045           the virtual system.
- 1046           • The virtual system is in the “Defined” virtual system state (that is, is not instantiated), which is  
1047           indicated by a value of 3 (Disabled) for the EnabledState property in the instance of the  
1048           CIM\_ComputerSystem class.

1049   The sequence of activities is as follows:

- 1050           1) The client resolves the CIM\_SettingsDefineState association to find the instance of the  
1051           CIM\_VirtualSystemSettingData class that is the head element of the “State” virtual system  
1052           configuration, invoking the intrinsic AssociatorNames( ) CIM operation with parameter values  
1053           set as follows:
- 1054           – The value of the ObjectName parameter refers to the instance of the  
1055           CIM\_ComputerSystem class that represents the virtual system.
  - 1056           – The value of the AssocClass parameter is set to “CIM\_SettingsDefineState”.
  - 1057           – The value of the ResultClass parameter is set to “CIM\_VirtualSystemSettingData”.
- 1058           The result of step 1) is a reference to an instance of the CIM\_VirtualSystemSettingData class.
- 1059           2) The client obtains instances of the CIM\_ElementSettingData association that reference the  
1060           result instance from step 1) to find the instance of the CIM\_VirtualSystemSettingData class that  
1061           is the head element of the “Defined” virtual system configuration, invoking the intrinsic  
1062           References( ) CIM operation with parameter values set as follows:
- 1063           – The value of the ObjectName parameter refers to the instance of the  
1064           CIM\_VirtualSystemSettingData class that is the head element of the “State” virtual system  
1065           configuration.
  - 1066           – The value of the ResultClass parameter is set to “CIM\_ElementSettingData”.
- 1067           The result of this step is a set of instances of the CIM\_ElementSettingData association. From  
1068           that set, the client selects the only instance where the IsDefault property has a value of 1 (Is  
1069           Default). The value of the SettingData property of that instance refers to the instance of the

- 1070 CIM\_VirtualSystemSettingData class that is the head element of the “Defined” virtual system  
1071 configuration.
- 1072 3) The client resolves the CIM\_VirtualSystemSettingDataComponent association to find instances  
1073 of the CIM\_ResourceAllocationSettingData class that represent resource allocation requests  
1074 within the “Defined” virtual system configuration, invoking the intrinsic Associators( ) CIM  
1075 operation with parameter values set as follows:
- 1076 – The value of the ObjectName parameter refers to the instance of the  
1077 CIM\_VirtualSystemSettingData class that is the head element of the “Defined” virtual  
1078 system configuration.
  - 1079 – The value of the AssocClass parameter is set to  
1080 “CIM\_VirtualSystemSettingDataComponent”.
  - 1081 – The value of the ResultClass parameter is set to “CIM\_ResourceAllocationSettingData”.
- 1082 The result of this step is a set of instances of the CIM\_ResourceAllocationSettingData class that  
1083 represent virtual resource allocation requests of the virtual system that are elements of the  
1084 “Defined” virtual system configuration.
- 1085 4) The client inspects the set of instances obtained in step 3), selecting only those instances  
1086 where the ResourceType property has a value of 4 (Memory).
- 1087 5) For each of the instances selected in step 4), the client then multiplies the of the VirtualQuantity  
1088 property with 1024, yielding the amount of virtual memory requested through the inspected  
1089 instance. The client builds the sum from the calculated values of all inspected instances with  
1090 step 5).

1091 NOTE: Subclause 7.1 requires the value of the AllocationUnits property to be “byte\*2<sup>10</sup>”. Alternatively the client  
1092 might inspect the value of the AllocationUnits property in order to determine the unit.

1093 **Result:** The client knows the amount of virtual memory requested for the virtual system.

1094 In the example CIM representation shown in Figure 5, the client initially would know the instance of the  
1095 CIM\_ComputerSystem class tagged VS that represents the virtual system. From there, the client would  
1096 follow the CIM\_SettingsDefineState association to the instance of the CIM\_VirtualSystemSettingData  
1097 class tagged STA\_VSSD. From there, the client would follow the CIM\_ElementSettingData association  
1098 where property IsDefault has a value of 1 (Is Default) to the instance of the  
1099 CIM\_VirtualSystemSettingData class tagged DEF\_VSSD. Finally, from there the client would follow the  
1100 CIM\_VirtualSystemSettingDataComponent association to obtain the instance of the  
1101 CIM\_ResourceAllocationSettingData class tagged MEM\_DEF that represents the only virtual memory  
1102 resource allocation request for the virtual system.

1103 The client would then multiply the value of the VirtualQuantity property with 1024, yielding a result of  
1104 1024\*1048576 bytes or 1048576 KB requested for the virtual systems virtual memory.

### 1105 9.2.3 Determine the Allocation Capabilities or Allocation Mutability

1106 **Assumption:** Any of the following:

- 1107 • The client knows a reference to an instance of the CIM\_System class that represents a host  
1108 system.
- 1109 • The client knows a reference to an instance of the CIM\_ResourcePool class that represents a  
1110 memory resource pool.
- 1111 • The client knows a reference to an instance of the CIM\_ResourceAllocationSettingData class  
1112 that represents a memory resource allocation request.
- 1113 • The client knows a reference to an instance of the CIM\_ResourceAllocationSettingData class  
1114 that represents a memory resource allocation.

1115 In the first two cases, this use case describes determining the related memory resource allocation  
1116 capabilities; in the latter two cases, this use case describes determining the related mutability.

1117 The sequence of activities is as follows:

1118 1) The client resolves the CIM\_ElementCapabilities association to find instances of the  
1119 CIM\_AllocationCapabilities class that represent allocation capabilities or mutability, invoking the  
1120 intrinsic Associators( ) CIM operation with parameter values set as follows:

- 1121 – The value of the ObjectName parameter refers to the input instance.
- 1122 – The value of the AssocClass parameter is set to “CIM\_ElementCapabilities”.
- 1123 – The value of the ResultClass parameter is set to “CIM\_AllocationCapabilities”.

1124 The result of this step is a set of instances of the CIM\_AllocationCapabilities class that  
1125 represent allocation capabilities or mutability.

1126 2) The client cycles through the set of instances of the CIM\_AllocationCapabilities class obtained  
1127 in step 1), fetching instances of the CIM\_SettingsDefineCapabilities association that reference  
1128 each of the instances from step 1) by invoking the intrinsic References( ) CIM operation with  
1129 parameter values set as follows:

- 1130 – The value of the ObjectName parameter each cycle refers another instance of the  
1131 CIM\_AllocationCapabilities class from the result set of step 1).
- 1132 – The value of the ResultClass parameter is set to “CIM\_SettingsDefineCapabilities”.

1133 The result of this step each time is a set of instances of the CIM\_SettingsDefineCapabilities  
1134 association.

1135 3) For each of the instances obtained in step 2), the client inspects the values of the ValueRole  
1136 and ValueRange properties to determine the type of limitation imposed by the instance of the  
1137 CIM\_ResourceAllocationSettingData class referenced by the value of the PartComponent  
1138 property in that association instance, as follows:

- 1139 – A default setting is designated through a value of 0 (Default) for the ValueRole property  
1140 and a value of 0 (Point) for the ValueRange property. A default setting does not apply for  
1141 the description of mutability.
- 1142 – A minimum setting is designated through a value of 3 (Supported) for the ValueRole  
1143 property and a value of 1 (Minimums) for the ValueRange property.
- 1144 – A maximum setting is designated through a value of 3 (Supported) for the ValueRole  
1145 property and a value of 2 (Maximums) for the ValueRange property.
- 1146 – An increment setting is designated through a value of 3 (Supported) for the ValueRole  
1147 property and a value of 3 (Increments) for the ValueRange property.

1148 4) The client obtains for each of the instances obtained in step 2) the referenced instance of the  
1149 CIM\_ResourceAllocationSettingData class, invoking the intrinsic GetInstance( ) CIM operation  
1150 with the value of the InstanceName parameter set to the value of the PartComponent property  
1151 from the association instance. That value refers to the instance of the  
1152 CIM\_ResourceAllocationSettingData class that represents the capabilities setting.

1153 The result is the instance of the CIM\_ResourceAllocationSettingData class with the values of all  
1154 non-null numeric properties that describe the settings.

1155 **Result:** The client knows the memory allocation capabilities of the system or the memory resource pool,  
1156 or the mutability of a memory resource allocation request or a memory resource allocation.

1157 In the example CIM representation shown in Figure 5, the capabilities of the system and the resource  
 1158 pool are represented by the instance of the CIM\_AllocationCapabilities class tagged CAP that is  
 1159 referenced likewise through an instance of the CIM\_ElementCapabilities association from the instance of  
 1160 the CIM\_System class tagged HOST that represents the system and from the instance of the  
 1161 CIM\_ResourcePool class tagged MEM\_POOL that represents a memory resource pool. Four instances of  
 1162 the CIM\_ResourceAllocationSettingData class tagged CAP\_DEF, CAP\_MIN, CAP\_MAX, and CAP\_INC  
 1163 describe the applicable default, minimum, maximum, and increment settings that describe the allocation  
 1164 capabilities. These instances are all associated through respectively parameterized instances of the  
 1165 CIM\_SettingsDefineCapabilities association with the instance of the CIM\_AllocationCapabilities class  
 1166 tagged CAP.

1167 In the example CIM representation shown in Figure 5, the mutability of the memory resource allocation  
 1168 request is represented by the instance of the CIM\_AllocationCapabilities class tagged DEF\_MUT. The  
 1169 mutability of the memory resource allocation is represented by the instance of the  
 1170 CIM\_AllocationCapabilities class tagged DEF\_STA. Values of elements in the value sets of the  
 1171 SupportedAddStates[ ] and SupportedRemoveStates[ ] array properties indicate the virtual system state  
 1172 for which respective additions and removals or memory resource allocation requests or memory resource  
 1173 allocations are supported. For both instances of the CIM\_AllocationCapabilities class, three instances of  
 1174 the CIM\_ResourceAllocationSettingData class are associated through respectively parameterized  
 1175 instances of the CIM\_SettingsDefineCapabilities association that describe minimum, maximum, and  
 1176 increment settings.

#### 1177 9.2.4 Determine the Default Memory Allocation Capabilities

1178 **Assumption:** The client knows a reference to the instance of the CIM\_System class that represents the  
 1179 host system.

1180 The sequence of activities is as follows:

- 1181 1) The client obtains instances of the CIM\_ElementCapabilities association that reference the  
 1182 instance of the CIM\_System class, invoking the intrinsic References( ) CIM operation with  
 1183 parameter values set as follows:
  - 1184 – The value of the ObjectName parameter refers to the instance of the CIM\_System class.
  - 1185 – The value of the ResultClass parameter is set to “CIM\_ElementCapabilities”.

1186 The result of this step is a set of instances of the CIM\_ElementCapabilities association.

- 1187 2) From the result set of step 1), the client drops those instances where the value set of the  
 1188 Characteristics[ ] array property does not contain an element with the value 2 (Default).

1189 The result of this step is a set of instances of the CIM\_ElementCapabilities association that  
 1190 reference instances of the CIM\_AllocationCapabilities class that represent the default allocation  
 1191 capabilities of the system for a number of resource types.

- 1192 3) For each of the association instances obtained in step 2), the client obtains the instance of the  
 1193 CIM\_AllocationCapabilities class that is referenced by the value of the Capabilities property in  
 1194 the respective association instance, invoking the intrinsic GetInstance( ) CIM operation with the  
 1195 value of the InstanceName parameter set to the value of the Capabilities property.

1196 The result of this step is a set of instances of the CIM\_ResourceAllocationSettingData class that  
 1197 represent the system's default allocation capabilities for a number of resource types.

- 1198 4) From the result set of step 3), the client drops those instances where the value set of the  
 1199 ResourceType property is not 4 (Memory).

1200 The result of this step is one instance of the CIM\_ResourceAllocationSettingData class that  
 1201 represents the system's default allocation capabilities for the memory resource type. The client  
 1202 continues as in use case 9.2.3 step 2) in order to determine the set of instances of the

1203 CIM\_ResourceAllocationSettingData that represent the settings for the default memory  
1204 resource allocation capabilities.

1205 **Result:** The client knows the default memory allocation capabilities of the system.

1206 In the example CIM representation shown in Figure 5, the default allocation capabilities for the memory  
1207 resource type of the system are represented by the instance of the CIM\_AllocationCapabilities class  
1208 tagged CAP.

### 1209 9.2.5 Determine the Default Memory Resource Pool

1210 **Assumption:** The client knows a reference to the instance of the CIM\_AllocationCapabilities class that  
1211 represents the default memory resource allocation capabilities of the system; see 9.2.4.

1212 The sequence of activities is as follows:

1213 1) The client obtains instances of the CIM\_ElementCapabilities association that reference the  
1214 instance of the CIM\_AllocationCapabilities class, invoking the intrinsic References( ) CIM  
1215 operation with parameter values set as follows:

1216 – The value of the ObjectName parameter refers to the instance of the  
1217 CIM\_AllocationCapabilities class.

1218 – The value of the ResultClass parameter is set to “CIM\_ElementCapabilities”.

1219 The result of this step is a set of instances of the CIM\_ElementCapabilities association.

1220 2) From the result set of step 1), the client drops those instances where the value set of the  
1221 Characteristics[ ] array property does not contain an element with the value 2 (Default).

1222 The result of this step is a set of two instances of the CIM\_ElementCapabilities association. One  
1223 association instance references the instance of the CIM\_ResourcePool class that represent the  
1224 default memory resource pool, and one instance references the instance of the CIM\_System  
1225 class that represents the host system.

1226 3) The client selects the instance of the CIM\_ElementCapabilities association from the result of  
1227 step 2) that references the instance of the CIM\_ResourcePool class by comparing the value of  
1228 the ManagedElement property against the known reference to the CIM\_System class that  
1229 represents the host system and dropping that association instance. The client uses the  
1230 remaining association instance from the result set of step 2) to obtain the instance of the  
1231 CIM\_ResourcePool class that is referenced by the value of the ManagedElement property in  
1232 that association instance, invoking the intrinsic GetInstance( ) CIM operation with the value of  
1233 the InstanceName parameter set to the value of the ManagedElement property.

1234 The result of this step is the instance of the CIM\_ResourcePool class that represents the  
1235 system’s default memory resource pool.

1236 **Result:** The client knows the default memory resource pool of the system.

1237 In the example CIM representation shown in Figure 5, the default memory resource pool is represented  
1238 by the instance of the CIM\_ResourcePool class tagged MEM\_POOL.

### 1239 9.2.6 Obtain the Memory Pool with the Largest Unreserved Capacity

1240 **Assumption:** The client knows a reference to the instance of the CIM\_System class that represents the  
1241 host system.

1242 The sequence of activities is as follows:

1243 1) The client resolves the CIM\_HostedPool association to find instances of the CIM\_ResourcePool  
1244 class that represent resource pools hosted by the host system, invoking the intrinsic  
1245 AssociatorNames( ) CIM operation with parameter values set as follows:

1246 – The value of the ObjectName parameter refers to the instance of the CIM\_System class  
1247 that represents the host.

1248 – The value of the AssocClass parameter is set to “CIM\_HostedPool”.

1249 – The value of the ResultClass parameter is set to “CIM\_ResourcePool”.

1250 The result of this step is a set of instances of the CIM\_ResourcePool class that represent  
1251 resource pools hosted by the host system.

1252 2) The client selects from the result set of step 1) only those instances where the value of the  
1253 ResourceType property is 4 (Memory).

1254 The result is a set of instances of the CIM\_ResourcePool class that represent memory resource  
1255 pools hosted by the host system.

1256 3) The client inspects the value of the Capacity and the Reserved properties in all instances  
1257 selected with step 2), and each time calculates the amount of unreserved memory capacity by  
1258 subtracting the value of the Reserved property from the value of the Capacity property.

1259 4) From all pools inspected in step 3), the client selects the one that has the largest free capacity.

1260 5) The client checks the resource pool selected in step 4) for architectural limitations as expressed  
1261 by the pool’s capabilities, applying use case 9.2.3.

1262 **Result:** The client knows the memory resource pool with the largest unreserved capacity.

1263 NOTE: The largest extent of memory actually available may be significantly smaller than indicated by the result  
1264 because fragmentation may subdivide the amount of memory available into several smaller extents.

1265 In the example CIM representation shown in Figure 5, the client initially would know the instance of the  
1266 CIM\_System class tagged HOST that represents the host system. From there, the client would follow the  
1267 CIM\_HostedPool association to instances of the CIM\_ResourcePool class. Typically the association  
1268 resolution would yield more than one instance, including instances that represent resource pools of other  
1269 resource types, such that the client is required to select only those instances where the value of the  
1270 ResourceType property is 4 (Memory). In Figure 5, there is only the instance of the CIM\_ResourcePool  
1271 class, tagged MEM\_POOL, so the selection process is not required. From that instance, the client takes  
1272 the value of the Capacity property and subtracts the value of the Reserved property (4194304 – 1048576)  
1273 KB, yielding 3145728 KB as the maximum amount of memory potentially available from the pool.

## 1274 10 CIM Elements

1275 Table 2 lists CIM elements that are defined or specialized for this profile. Each CIM element shall be  
1276 implemented as described in Table 2. The CIM Schema descriptions for any referenced element and its  
1277 sub-elements apply.

1278 Classes 7 (“Implementation”) and 8 (“Methods”) may impose additional requirements on these elements.

1279

Table 2 – CIM Elements: Memory Resource Virtualization Profile

Element	Requirement	Description
<b>Classes</b>		
CIM_AffectedJobElement	Conditional	See 10.1.
CIM_AllocationCapabilities (Capabilities)	Mandatory	See 10.2.
CIM_AllocationCapabilities (Mutability)	Optional	See 10.3.
CIM_Component (Memory Composition)	Optional	See 10.4.
CIM_Component (Resource Pool)	Optional	See 10.5.
CIM_ConcreteJob	Conditional	See 10.6.
CIM_ElementAllocatedFromPool	Mandatory	See 10.7.
CIM_ElementCapabilities (Capabilities)	Mandatory	See 10.8.
CIM_ElementCapabilities (Mutability)	Conditional	See 10.9.
CIM_ElementCapabilities (Resource Pool)	Mandatory	See <a href="#">DSP1041</a> .
CIM_ElementSettingData (Memory Pool)	Mandatory	See 10.10.
CIM_ElementSettingData (Memory Resource)	Mandatory	See 10.11.
CIM_HostedDependency	Optional	See 10.12.
CIM_HostedResourcePool	Mandatory	See <a href="#">DSP1041</a> .
CIM_HostedService	Mandatory	See <a href="#">DSP1041</a> .
CIM_Memory (Host System)	Conditional	See 10.13.
CIM_Memory (Virtual System)	Mandatory	See 10.14.
CIM_RegisteredProfile	Mandatory	See 10.15.
CIM_ResourceAllocationFromPool	Optional	See 10.16.
CIM_ResourceAllocationSettingData	Mandatory	See 10.17.
CIM_ResourcePool	Mandatory	See 10.18.
CIM_ResourcePoolConfigurationCapabilities	Mandatory	See 10.19.
CIM_ResourcePoolConfigurationService	Mandatory	See <a href="#">DSP1041</a> .
CIM_SettingsDefineState	Mandatory	See 10.20.
CIM_ServiceAffectsElement	Mandatory	See 10.21.
CIM_SystemDevice (Virtual Memory)	Mandatory	See 10.22.
CIM_SystemDevice (Host Memory)	Optional	See 10.23.
<b>Indications</b>		
None defined		

1280 **10.1 CIM\_AffectedJobElement**

1281 The support of the CIM\_AffectedJobElement class is conditional.

1282 Conditional Requirement: The CIM\_AffectedJobElement association shall be supported if asynchronous  
1283 operations for resource pool management are supported; see 7.2.2.8.4.



1284 If the CIM\_AffectedJobElement association is supported, instances of the CIM\_AffectedJobElement  
 1285 association shall associate an instance of the CIM\_ConcreteJob class that represents an asynchronous  
 1286 memory resource pool management task and all of the following:

- 1287 • the instance of the CIM\_ResourcePool class that represents a memory resource pool that is  
 1288 affected by the asynchronous memory resource pool management task
- 1289 • the instance of the CIM\_Memory class that represents host memory that is affected by the  
 1290 asynchronous memory resource pool management task

1291 Table 3 lists the requirements for elements of this association. These requirements are in addition to  
 1292 those specified in the CIM Schema and in [DSP1041](#).

1293 **Table 3 – Association: CIM\_AffectedJobElement**

Elements	Requirement	Notes
AffectedElement	Mandatory	<b>Key:</b> Value shall reference an instance of the CIM_ResourcePool class or the CIM_Memory class. <b>Cardinality:</b> *
AffectingElement	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ConcreteJob class. <b>Cardinality:</b> *

1294 **10.2 CIM\_AllocationCapabilities (Capabilities)**

1295 See 7.3.1.1 for detailed implementation requirements for this class if it is used for the representation of  
 1296 memory resource allocation capabilities of systems or of memory resource pools.

1297 Table 4 lists the requirements for elements of this class in this case. These requirements are in addition  
 1298 to those specified in the CIM Schema and in [DSP1043](#).

1299 **Table 4 – Class: CIM\_AllocationCapabilities (Memory Allocation Capabilities)**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
RequestTypesSupported	Mandatory	See 7.3.1.1.2.
SharingMode	Mandatory	See 7.3.1.1.3.
SupportedAddStates[ ]	Mandatory	Value shall be NULL.
SupportedRemoveStates[ ]	Mandatory	Value shall be NULL.

1300 **10.3 CIM\_AllocationCapabilities (Mutability)**

1301 The support of the CIM\_AllocationCapabilities class for the representation of the mutability of memory  
 1302 resource allocation requests and memory resource allocations is optional.

1303 If the CIM\_AllocationCapabilities class is supported for the representation of the mutability of memory  
 1304 resource allocation requests and memory resource allocations, see 7.3.2.2 for detailed implementation  
 1305 requirements.

1306 Table 5 lists the requirements for elements of this class. These requirements are in addition to those  
 1307 specified in the CIM Schema and in [DSP1043](#).

1308 **Table 5 – Class: CIM\_AllocationCapabilities (Memory Allocation Mutability)**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
RequestTypesSupported	Mandatory	See 7.3.2.2.1.
SharingMode	Mandatory	See 7.3.2.2.2.
SupportedAddStates[ ]	Optional	See 7.3.2.2.3.
SupportedRemoveStates[ ]	Optional	See 7.3.2.2.4.

#### 1309 10.4 CIM\_Component (Memory Composition)

1310 The support of the CIM\_Component association for the representation of memory compositions is  
 1311 optional.

1312 If the CIM\_Component association is supported for the representation of memory compositions that are  
 1313 composed of other memory extents, instances of an association that is based on the CIM\_Component  
 1314 association shall associate an instance of the CIM\_Memory class that represents the memory  
 1315 composition and any instance of the CIM\_Memory class that represent composing memory extents. If an  
 1316 implementation does not implement a more specific association based on the CIM\_Component  
 1317 association, the CIM\_ConcreteComponent association should be implemented.

1318 NOTE: The CIM\_Component association is abstract; therefore it cannot be directly implemented. On the other  
 1319 hand, clients may directly follow abstract associations.

1320 Table 6 lists the requirements for elements of this association. These requirements are in addition to  
 1321 those specified in the CIM Schema.

1322 **Table 6 – Association: CIM\_Component (Memory Resource)**

Elements	Requirement	Notes
GroupComponent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_Memory class that represents the memory composition. <b>Cardinality:</b> 0..1
PartComponent	Mandatory	<b>Key:</b> Value shall reference an instance of the CIM_Memory class that represents a composing memory extent. <b>Cardinality:</b> *

#### 1323 10.5 CIM\_Component (Resource Pool)

1324 The support of the CIM\_Component association for the representation of memory extents that are  
 1325 aggregated by resource pools is optional.

1326 If the CIM\_Component association is supported for the representation of memory extents that are  
 1327 aggregated by resource pools, instances of an association that is based on the CIM\_Component  
 1328 association shall associate an instance of the CIM\_ResourcePool class that represents a primordial  
 1329 memory resource pool and any instance of the CIM\_Memory class that represents host memory that is  
 1330 aggregated into the pool. If an implementation does not implement a more specific association based on  
 1331 the CIM\_Component association, the CIM\_ConcreteComponent association should be implemented.

1332 NOTE: The CIM\_Component association is abstract; therefore it cannot be directly implemented. On the other  
 1333 hand, clients may directly follow abstract associations.

1334 Table 7 lists the requirements for elements of this association. These requirements are in addition to  
 1335 those specified in the CIM Schema and in [DSP1041](#).

1336 **Table 7 – Association: CIM\_Component (Resource Pool)**

Elements	Requirement	Notes
GroupComponent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourcePool class that represents a memory resource pool. <b>Cardinality:</b> 0..1
PartComponent	Mandatory	<b>Key:</b> Value shall reference an instance of the CIM_Memory class that represents host memory aggregated into the pool. <b>Cardinality:</b> *

1337 **10.6 CIM\_ConcreteJob**

1338 The support of the CIM\_ConcreteJob class is conditional.

1339 Conditional Requirement: The CIM\_ConcreteJob class shall be supported if asynchronous operations for  
 1340 resource pool management are supported; see 7.2.2.8.4.

1341 If the CIM\_ConcreteJob is supported, instances of the CIM\_ConcreteJob class shall represent  
 1342 asynchronous memory resource pool management tasks.

1343 Table 8 lists the requirements for elements of this class. These requirements are in addition to those  
 1344 specified in the CIM Schema and in [DSP1041](#).

1345 **Table 8 – Class: CIM\_ConcreteJob**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
DeleteOnCompletion	Mandatory	Value shall be TRUE.
ElementName	Mandatory	Value shall conform to pattern ".*"
ErrorCode	Mandatory	None
ErrorDescription	Mandatory	None
JobStatus	Mandatory	None
JobState	Mandatory	None

1346 **10.7 CIM\_ElementAllocatedFromPool**

1347 An instance of the CIM\_ElementAllocatedFromPool association shall associate an instance of the  
 1348 CIM\_ResourcePool class that represents a memory resource pool with each instance of the CIM\_Memory  
 1349 class that represents virtual memory resulting from a memory resource allocation out of the pool.

1350 Table 9 lists the requirements for elements of this association. These requirements are in addition to  
 1351 those specified in the CIM Schema and in [DSP1041](#).

1352 **Table 9 – Association: CIM\_ElementAllocatedFromPool**

Elements	Requirement	Notes
Antecedent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourcePool class that represents a memory resource pool. <b>Cardinality:</b> 1
Dependent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_Memory class that represents virtual memory resulting from a memory allocation from the pool. <b>Cardinality:</b> *

1353 **10.8 CIM\_ElementCapabilities (Capabilities)**

1354 Instances of the CIM\_ElementCapabilities association shall associate all of the following:

- 1355 • the instance of the CIM\_System class that represents the host system with each instance of the  
 1356 CIM\_AllocationCapabilities class that represents memory allocation capabilities of the host  
 1357 system
- 1358 • an instance of the CIM\_ResourcePool that represents a memory resource pool with each  
 1359 instance of the CIM\_AllocationCapabilities class that represents memory allocation capabilities  
 1360 of the memory resource pool

1361 Table 10 lists the requirements for elements of this class. These requirements are in addition to those  
 1362 specified in the CIM Schema and in [DSP1043](#).

1363 **Table 10 – Association: CIM\_ElementCapabilities (Capabilities)**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> Host: See 7.3.1.2 and 7.3.1.3. Pool: See 7.3.1.4 and 7.3.1.5. <b>Cardinality:</b> *
Capabilities	Mandatory	<b>Key:</b> Host: See 7.3.1.2 and 7.3.1.3. Pool: See 7.3.1.4 and 7.3.1.5. <b>Cardinality:</b> *
Characteristics	Mandatory	Host: See 7.3.1.2 and 7.3.1.3. Pool: See 7.3.1.4 and 7.3.1.5.

1364 **10.9 CIM\_ElementCapabilities (Mutability)**

1365 The support of the CIM\_ElementCapabilities association for the representation of the mutability of a  
 1366 memory resource allocation request or a memory resource allocation is conditional.

1367 Conditional Requirement: The support is required if the CIM\_AllocationCapabilities class is supported for  
 1368 the representation of the mutability of memory resource allocation requests or memory resource  
 1369 allocations; see 10.3.

1370 If the CIM\_ElementCapabilities association is supported for the representation of the mutability of a  
 1371 memory resource allocation request or a memory resource allocation, an instance of the  
 1372 CIM\_ElementCapabilities association shall associate an instance of the  
 1373 CIM\_ResourceAllocationSettingData class that represents a memory resource allocation or memory  
 1374 resource allocation request with each instance of the CIM\_AllocationCapabilities class that represents the  
 1375 mutability of the memory resource allocation or memory resource allocation request.

1376 Table 11 lists the requirements for elements of this class. These requirements are in addition to those  
 1377 specified in the CIM Schema and in [DSP1043](#).

1378 **Table 11 – Association: CIM\_ElementCapabilities (Mutability)**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourceAllocationSettingData class. <b>Cardinality:</b> *
Capabilities	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_AllocationCapabilities class. <b>Cardinality:</b> *
Characteristics[ ]	Mandatory	Value shall be { 3 (Current) }.

1379 **10.10 CIM\_ElementSettingData (Memory Resource Pool)**

1380 An instance of the CIM\_ElementSettingData association shall associate an instance of the  
 1381 CIM\_ResourcePool class that represents a concrete memory resource pool and the instance of the  
 1382 CIM\_ResourceAllocationSettingData class that represents the memory resource allocation that describes  
 1383 the allocation of the concrete resource pool from another resource pool.

1384 Table 12 lists the requirements for elements of this class. These requirements are in addition to those  
 1385 specified in the CIM Schema and in [DSP1041](#).

1386 **Table 12 – Association: CIM\_ElementSettingData (Memory Resource Pool)**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourcePool class that represents the concrete memory resource pool. <b>Cardinality:</b> 0..1
SettingData	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation. <b>Cardinality:</b> 0..1

1387 **10.11 CIM\_ElementSettingData (Memory Resource)**

1388 An instance of the CIM\_ElementSettingData association shall associate an instance of the  
 1389 CIM\_ResourceAllocationSettingData class that represents a memory resource allocation and the instance  
 1390 of the CIM\_ResourceAllocationSettingData class that represents the corresponding memory resource  
 1391 allocation request.

1392 Table 13 lists the requirements for elements of this class. These requirements are in addition to those  
 1393 specified in the CIM Schema and in [DSP1041](#).

1394 **Table 13 – Association: CIM\_ElementSettingData (Memory Resource)**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation.  <b>Cardinality:</b> 1
SettingData	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents the memory resource allocation request.  <b>Cardinality:</b> 0..1
IsDefault	Mandatory	Value shall be 1 (Is Default).
IsCurrent	Mandatory	Unspecified.
IsNext	Mandatory	Unspecified.

1395 **10.12 CIM\_HostedDependency**

1396 The support of the CIM\_HostedDependency association is optional.

1397 If the CIM\_HostedDependency association is supported, an instance of the CIM\_HostedDependency  
 1398 association shall associate an instance of the CIM\_Memory class that represents virtual memory with the  
 1399 instance of the CIM\_Memory class that represents host memory that is dedicated for the support of the  
 1400 virtual memory.

1401 Table 14 lists the requirements for elements of this association. These requirements are in addition to  
 1402 those specified in the CIM Schema and in [DSP1041](#).

1403 **Table 14 – Association: CIM\_HostedDependency**

Elements	Requirement	Notes
Antecedent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_Memory class that represents host memory.  <b>Cardinality:</b> 0..1
Dependent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_Memory class that represents virtual memory.  <b>Cardinality:</b> 0..1

1404 **10.13 CIM\_Memory (Host System)**

1405 The support of the CIM\_Memory class for the representation of host memory is conditional.

1406 Conditional Requirement: The support is required if the CIM\_SystemDevice association is supported for  
 1407 the representation of host memory; see 10.23.

1408 Table 15 lists the requirements for elements of this class. These requirements are in addition to those  
 1409 specified in the CIM Schema and in [DSP1026](#) if that is implemented.

1410 **Table 15 – Class: CIM\_Memory (Host System)**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	<b>Key</b>
CreationClassName	Mandatory	<b>Key</b>
SystemName	Mandatory	<b>Key</b>
Name	Mandatory	<b>Key</b>
EnabledState	Mandatory	Unspecified
RequestedState	Mandatory	Unspecified
StartingAddress	Mandatory	Unspecified
EndingAddress	Mandatory	Unspecified

1411 **10.14 CIM\_Memory (Virtual System)**

1412 See 7.2.5 for detailed implementation requirements for this class if it is used for the representation of  
 1413 virtual memory or virtual memory composition.

1414 Table 16 lists the requirements for elements of this class. These requirements are in addition to those  
 1415 specified in the CIM Schema and in [DSP1026](#) if that is implemented.

1416 **Table 16 – Class: CIM\_Memory (Virtual System)**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	<b>Key</b>
CreationClassName	Mandatory	<b>Key</b>
SystemName	Mandatory	<b>Key</b>
Name	Mandatory	<b>Key</b>
EnabledState	Mandatory	Unspecified
RequestedState	Mandatory	Unspecified
StartingAddress	Mandatory	Unspecified
EndingAddress	Mandatory	Unspecified

1417 **10.15 CIM\_RegisteredProfile**1418 The use of the CIM\_RegisteredProfile class is specified by [DSP1033](#).1419 Table 17 lists the requirements for elements of this class. These requirements are in addition to those  
1420 specified in [DSP1033](#).1421 **Table 17 – Class: CIM\_RegisteredProfile**

Elements	Requirement	Notes
RegisteredOrganization	Mandatory	Value shall be set to 2 (DMTF).
RegisteredName	Mandatory	Value shall be set to “Memory Resource Virtualization”.
RegisteredVersion	Mandatory	Value shall be set to the version of this profile: “1.0.0”.

1422 **10.16 CIM\_ResourceAllocationFromPool**

1423 The support of the CIM\_ResourceAllocationFromPool association is optional.

1424 If the CIM\_ResourceAllocationFromPool association is supported, an instance of the  
1425 CIM\_ResourceAllocationFromPool association shall associate an instance of the CIM\_ResourcePool  
1426 class that represents a memory resource pool with each instance of the  
1427 CIM\_ResourceAllocationSettingData class that represents a memory resource allocation from the pool.1428 Table 18 lists the requirements for elements of this association. These requirements are in addition to  
1429 those specified in the CIM Schema and in [DSP1041](#).1430 **Table 18 – Association: CIM\_ResourceAllocationFromPool**

Elements	Requirement	Notes
Antecedent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourcePool class that represents a memory resource pool. <b>Cardinality:</b> 0..1
Dependent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourceAllocationSettingData class that represents a memory resource allocation from the pool. <b>Cardinality:</b> *

1431 **10.17 CIM\_ResourceAllocationSettingData**

1432 See 7.3.2.2 for detailed implementation requirements for this class.

1433 Table 19 lists the requirements for elements of this class. These requirements are in addition to those  
1434 specified in the CIM Schema and in [DSP1041](#).



1435

**Table 19 – Class: CIM\_ResourceAllocationSettingData**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b> ; see <a href="#">DSP1041</a> .
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See <a href="#">DSP1041</a> .
PoolID	Mandatory	See 7.2.4.2.
ConsumerVisibility	Optional	See 7.2.4.3.
HostResource[ ]	Optional	See 7.2.4.4.
AllocationUnits	Mandatory	See 7.1.
VirtualQuantity	Mandatory	See 7.2.4.5.
Reservation	Optional	See 7.2.4.6.
Limit	Optional	See 7.2.4.7.
Weight	Optional	See 7.2.4.8.
AutomaticAllocation	Optional	See <a href="#">DSP1041</a> .
AutomaticDeallocation	Optional	See <a href="#">DSP1041</a> .
Parent	Optional	See 7.2.4.9.
Connection[ ]	Optional	See 7.2.4.10.
MappingBehavior	Optional	See 7.2.4.11.

1436 **10.18 CIM\_ResourcePool**

1437 Instances of the CIM\_ResourcePool class shall represent memory resource pools.

1438 Table 20 lists the requirements for elements of this class. These requirements are in addition to those  
 1439 specified in the CIM Schema and in [DSP1041](#).

1440

**Table 20 – Class: CIM\_ResourcePool**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
ElementName	Optional	See <a href="#">DSP1041</a> .
PoolID	Mandatory	See 7.2.2.1.
Primordial	Mandatory	See 7.2.2.2.
Capacity	Conditional	See 7.2.2.5.
Reserved	Optional	See 7.2.2.4.
ResourceType	Mandatory	Value shall be 4 (Memory).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See <a href="#">DSP1041</a> .
AllocationUnits	Mandatory	See 7.1.

1441 **10.19 CIM\_ResourcePoolConfigurationCapabilities**

1442 An instance of the CIM\_ResourcePoolConfigurationCapabilities class shall represent the capabilities of a  
1443 memory resource pool configuration service.

1444 Table 21 lists the requirements for elements of this class. These requirements are in addition to those  
1445 specified in the CIM Schema and in [DSP1041](#).

1446 **Table 21 – Class: CIM\_ResourcePoolConfigurationCapabilities**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
AsynchronousMethodsSupported[ ]	Mandatory	See 7.2.2.8.
SynchronousMethodsSupported[ ]	Mandatory	See 7.2.2.8.

1447 **10.20 CIM\_SettingsDefineState**

1448 An instance of the CIM\_SettingsDefineState association shall associate an instance of the CIM\_Memory  
1449 class that represents virtual memory and the instance of the CIM\_ResourceAllocationSettingData class  
1450 that represents the memory resource allocation that yields the virtual memory.

1451 Table 22 lists the requirements for elements of this association. These requirements are in addition to  
1452 those specified in the CIM Schema and in [DSP1041](#).

1453 **Table 22 – Association: CIM\_SettingsDefineState**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> Value shall reference an instance of the CIM_Memory class. <b>Cardinality:</b> 0..1
SettingData	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourceAllocationSettingData class. <b>Cardinality:</b> 0..1

1454 **10.21 CIM\_ServiceAffectsElement**

1455 An instance of the CIM\_ServiceAffectsElement association shall associate an instance of the  
1456 CIM\_ResourcePoolConfigurationService class that represents a memory resource pool configuration  
1457 service and each instance of the CIM\_ResourcePool class that represents a memory resource pool that  
1458 is configurable through the service.

1459 Table 23 lists the requirements for elements of this association. These requirements are in addition to  
1460 those specified in the CIM Schema and in [DSP1041](#).

1461 **Table 23 – Association: CIM\_ServiceAffectsElement**

Elements	Requirement	Notes
AffectedElement	Mandatory	<b>Key:</b> Value shall reference an instance of the CIM_ResourcePool class. <b>Cardinality:</b> *

Elements	Requirement	Notes
AffectingElement	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_ResourcePoolConfigurationService class. <b>Cardinality:</b> 1

1462 **10.22 CIM\_SystemDevice (Virtual Memory)**

1463 An instance of the CIM\_SystemDevice association shall associate the instance of the  
1464 CIM\_ComputerSystem class that represents a virtual system and each instance of the CIM\_Memory  
1465 class that represents virtual memory in scope of the virtual system.

1466 Table 24 lists the requirements for elements of this association. These requirements are in addition to  
1467 those specified in the CIM Schema and in [DSP1041](#).

1468 **Table 24 – Association: CIM\_SystemDevice (Virtual Memory)**

Elements	Requirement	Notes
GroupComponent	Mandatory	<b>Key:</b> Value shall reference an instance of the CIM_System class. <b>Cardinality:</b> 1
PartComponent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_Memory class. <b>Cardinality:</b> *

1469 **10.23 CIM\_SystemDevice (Host Memory)**

1470 Support of the CIM\_SystemDevice association for the representation of host memory is optional; see  
1471 7.2.1.

1472 NOTE: Support is mandatory if [DSP1026](#) is implemented for the host system.

1473 If the CIM\_SystemDevice association is supported for the representation of host memory, an instance of  
1474 the CIM\_SystemDevice association shall associate the instance of the CIM\_System class that represents  
1475 the scoping host system and each instance of the CIM\_Memory class that represents host memory in  
1476 scope of the scoping host system.

1477 Table 25 lists the requirements for elements of this association. These requirements are in addition to  
1478 those specified in the CIM Schema, in [DSP1041](#), and in [DSP1026](#) if that is implemented.

1479 **Table 25 – Association: CIM\_SystemDevice (Host Memory)**

Elements	Requirement	Notes
GroupComponent	Mandatory	<b>Key:</b> Value shall reference an instance of the CIM_System class. <b>Cardinality:</b> 1
PartComponent	Mandatory	<b>Key:</b> Value shall reference the instance of the CIM_Memory class. <b>Cardinality:</b> *

**ANNEX A  
(Informative)****Change Log**1480  
1481  
1482  
1483

1484

Version	Date	Description
1.0.0	2009-07-14	DMTF Standard Release

1485