



1
2
3
4

Document Number: DSP1044

Date: 2010-04-22

Profile Version: 1.0.0

5 **Processor Resource Virtualization Profile**

6 **Document Type: Specification**
7 **Document Status: DMTF Standard**
8 **Document Language: E**

9 Copyright Notice

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

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

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25 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83

CONTENTS

1	Scope	7
2	Normative references	7
3	Terms and definitions	8
4	Symbols and abbreviated terms	10
5	Synopsis	10
6	Description	11
6.1	General	11
6.2	Processor resource virtualization class schema	11
6.3	Resource pools	13
6.4	Resource allocation	14
7	Implementation	15
7.1	Allocation units	15
7.2	Host resources	17
7.3	Resource pools	17
7.4	Aggregation of host resources	18
7.5	Processor resource pool hierarchies	19
7.6	Default processor resource pool	19
7.7	Processor resource pool management	19
7.8	Processor resource allocation	19
7.9	Virtual processor	23
8	Methods	24
8.1	Profile conventions for operations	24
8.2	CIM_Processor for host processors	24
8.3	CIM_Processor for virtual processor	24
8.4	CIM_ReferencedProfile	24
8.5	CIM_RegisteredProfile	24
8.6	CIM_SystemDevice for host processors	24
8.7	CIM_SystemDevice for virtual processors	25
9	Use cases	25
9.1	Use case 1 – Increase the allocated processor capacity to a virtual machine in MHz	28
9.2	Use case 2 – Increase the allocated processor capacity to a virtual machine in percent	28
9.3	Use case 3 - Add a virtual processor to a virtual system	29
9.4	Use case 4 – Allocation of processor resource by weight	31
9.5	Use case 5 – Allocation of an additional processor resource	32
10	CIM elements	34
10.1	CIM_Component for resource pool	35
10.2	CIM_ElementAllocatedFromPool for allocated virtual processors	36
10.3	CIM_ElementAllocatedFromPool for resource pool hierarchies	36
10.4	CIM_ElementSettingData for processor resource allocation	37
10.5	CIM_ElementSettingData for processor resource pool	37
10.6	CIM_HostedDependency	38
10.7	CIM_Processor (host processor)	38
10.8	CIM_Processor (virtual processor)	38
10.9	CIM_RegisteredProfile	39
10.10	CIM_ResourceAllocationFromPool	39
10.11	CIM_ResourceAllocationSettingData	39
10.12	CIM_ResourcePool	40
10.13	CIM_SettingsDefineState	41
10.14	CIM_SystemDevice for host processor	41
10.15	CIM_SystemDevice for virtual processor	42
ANNEX A (Informative)	Change Log	43

84 **Figures**

85	Figure 1 – Processor Resource Virtualization Profile: Class Diagram	12
86	Figure 2 – Processor Resource Virtualization Profile: Instance diagram	26
87	Figure 3 – Defined state	27
88	Figure 4 – Active state	27
89	Figure 5 – CIM_ModifyResourceSettings – Before	29
90	Figure 6 – RASD to Modify Resources	30
91	Figure 7 – CIM_ModifyResourceSettings - After	31
92	Figure 8 – CIM_AddResourceSettings - Before.....	32
93	Figure 9 – RASD to add processor	33
94	Figure 10 – CIM_AddResourceSettings - After.....	34
95		

96 **Tables**

	Table 1 – Related profiles	11
	Table 2 – Acronyms for RASD adapted for the representation of various flavors of allocation data.....	20
	Table 3 – CIM Elements: Processor Resource Virtualization Profile.....	34
	Table 4 – Association: CIM_Component for resource pool	36
	Table 5 – Association: CIM_ElementAllocatedFromPool	36
	Table 6 – Association: CIM_ElementSettingData	37
	Table 7 – Association: CIM_ElementSettingData for processor resource allocation	37
	Table 8 – Association: CIM_ElementSettingData (Processor Resource Pool)	37
	Table 9 – Association: CIM_HostedDependency	38
	Table 10 – Class: CIM_Processor (host processor)	38
	Table 11 – Class: CIM_Processor (virtual system).....	39
	Table 13 – Class: CIM_RegisteredProfile.....	39
	Table 14 – Association: CIM_ResourceAllocationFromPool	39
	Table 15 – Class: CIM_ResourceAllocationSettingData	40
	Table 16 – Class: CIM_ResourcePool.....	40
	Table 17 – Association: CIM_SettingsDefineState	41
	Table 18 – Association: CIM_SystemDevice (Host Processor).....	41
	Table 19 – Association: CIM_SystemDevice (Virtual Processor)	42

97

Foreword

98 This profile was prepared by the Server Virtualization Partitioning and Clustering workgroup of the DMTF.

99 The SVPC work group acknowledges the following people for their contributions to the development this
100 profile.

101 Editors:

- 102 • Michael Johanssen – IBM
- 103 • Lawrence Lamers – VMware Inc.
- 104 • Carl Waldspurger - VMware Inc.

105 Contributors:

- 106 • Gareth Bestor – IBM
- 107 • Ron Goering – IBM
- 108 • Daniel Hiltgen –VMware
- 109 • Ron Doyle – IBM
- 110 • Rene Schmidt – VMware Inc.
- 111 • Steffen Gararup – VMware Inc.
- 112 • Hemal Shah – Broadcom
- 113 • Fred Maciel – Hitachi Ltd.
- 114 • Lawrence Lamers – VMware Inc.
- 115 • Andreas Maier – IBM
- 116 • John Parchem – Microsoft Corporation
- 117 • George Ericson – EMC
- 118 • Oliver Benke – IBM
- 119 • John Leung – Intel Corporation
- 120 • James Fehlig – Novell
- 121 • Nihar Shah – Microsoft Corporation
- 122 • Shishir Pardikar – Citrix Systems Inc.
- 123 • Stephen Schmidt – IBM
- 124 • Mark Hapner – Sun Microsystems
- 125 • Dave Barrett – Emulex
- 126 • John Suit – Fortisphere
- 127 • Jeff Wheeler – Cisco
- 128 • Mark Johnson – IBM

129

Introduction

130 The information in this specification should be sufficient for a provider or consumer of this data to
131 unambiguously identify the classes, properties, methods, and values that shall be instantiated and
132 manipulated to represent and manage a basic server and subsystems that are modeled using the DMTF
133 Common Information Model (CIM) core and extended model definitions.

134 The target audience for this specification is implementers who are writing CIM-based providers or
135 consumers of management interfaces that represent the components described in this document.

136

Processor Resource Virtualization Profile

137 1 Scope

138 This profile is a component profile that extends the management capabilities of the specialized profiles by
139 adding the support to represent and manage the allocation of processor resources to virtual systems.

140 2 Normative references

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

144 DMTF DSP0004, *CIM Infrastructure Specification 2.5*
145 http://www.dmtf.org/standards/published_documents/DSP0004_2.5.pdf

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

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

150 DMTF DSP1022, *CPU profile 1.0*
151 http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf

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

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

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

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

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

162 DMTF DSP1059, *Generic Device Resource Virtualization profile 1.0*
163 http://www.dmtf.org/standards/published_documents/DSP1059_1.0.pdf

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

166 **3 Terms and definitions**

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

169 **3.1**

170 **can**

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

172 **3.2**

173 **cannot**

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

175 **3.3**

176 **conditional**

177 indicates requirements to be followed strictly to conform to the document when the specified conditions
178 are met

179 **3.4**

180 **mandatory**

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

183 **3.5**

184 **may**

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

186 **3.6**

187 **need not**

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

189 **3.7**

190 **optional**

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

192 **3.8**

193 **referencing profile**

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

196 **3.9**

197 **shall**

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

200 **3.10**

201 **shall not**

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

204 **3.11**

205 **should**

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

- 208 **3.12**
209 **should not**
210 indicates that a certain possibility or course of action is deprecated but not prohibited
- 211 **3.13**
212 **unspecified**
213 indicates that this profile does not define any constraints for the referenced CIM element or operation
- 214 **3.14**
215 **client**
216 an application that exploits facilities specified by this profile
- 217 **3.15**
218 **implementation**
219 a set of CIM providers that realize the classes specified by this profile
- 220 **3.16**
221 **this profile**
222 this DMTF management profile – the Processor Resource Virtualization profile
- 223 **3.17**
224 **host processor resource**
225 host processor resources are processor devices or computing resource contained by the host system that
226 may be allocated with either exclusive or shared access to provide processing resources to a processor
227 resource pool or a virtual system.
- 228 **3.18**
229 **host system**
230 The scoping system containing resources that may be allocated and/or virtualized.
- 231 **3.19**
232 **processor resource**
233 a processor device or computing resource as seen by a consumer
- 234 **3.20**
235 **processor resource allocation**
236 the allocation of a processor resource from a processor resource pool to a virtual system
- 237 **3.21**
238 **processor resource allocation request**
239 a request for a processor resource allocation
- 240 **3.22**
241 **processor resource pool**
242 a resource pool that represents processor resources available for processor resource allocation
- 243 **3.23**
244 **processor resource pool configuration service**
245 a configuration service that supports the addition or removal of host storage processor resources to or
246 from a processor resource pool, and the creation or deletion of concrete subpools of a processor
247 resource pool

248 **3.24**
249 **virtual computer system**
250 the concept of a virtual system as applied to a computer system.
251 Other common industry terms are virtual machine, hosted computer, child partition, logical partition,
252 domain, guest, and container.#

253 **3.25**
254 **virtual processor**
255 the instantiation of the allocated host processor resources that is exposed to a virtual system via a logical
256 processor device.

257 **4 Symbols and abbreviated terms**

258 The following abbreviations are used in this document.

259 **4.1**
260 **CIM**
261 Common Information Model

262 **4.2**
263 **CIMOM**
264 CIM object manager

265 **4.3**
266 **RASD**
267 CIM_ResourceAllocationSettingData

268 **4.4**
269 **VS**
270 virtual system

271 **5 Synopsis**

272 **Profile Name:** Processor Resource Virtualization

273 **Version:** 1.0.0

274 **Organization:** DMTF

275 **CIM schema version:** 2.21

276 **Central Class:** CIM_ResourcePool

277 **Scoping Class:** CIM_System

278 This profile is a component profile that defines the minimum object model needed to provide for the CIM
279 representation and management of the virtualization of processors.

280 Table 1 lists other profiles that this profile depends on, or that may be used in context of this profile.

281

Table 1 – Related profiles

Profile Name	Organization	Version	Relationship	Description
<i>Resource Allocation</i>	DMTF	1.1	Specializes	The profile that adds the capability to represent the allocation of resources to consumers. See DSP1041 .
<i>Allocation Capabilities</i>	DMTF	1.0	Specializes	The profile that describes the default property values, supported property values, and range of property values for a resource allocation request. See DSP1043 .
<i>Profile Registration</i>	DMTF	1.0	Mandatory	The profile that specifies registered profiles. See DSP1033 .
<i>CPU</i>	DMTF	1.0	Optional	The profile that adds the capability to represent processors in a managed system. See 7.2. See DSP1022 .

282 The CPU Profile lists additional related DMTF management profiles; these relationships are not further
 283 specified in this profile.

284 **6 Description**

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

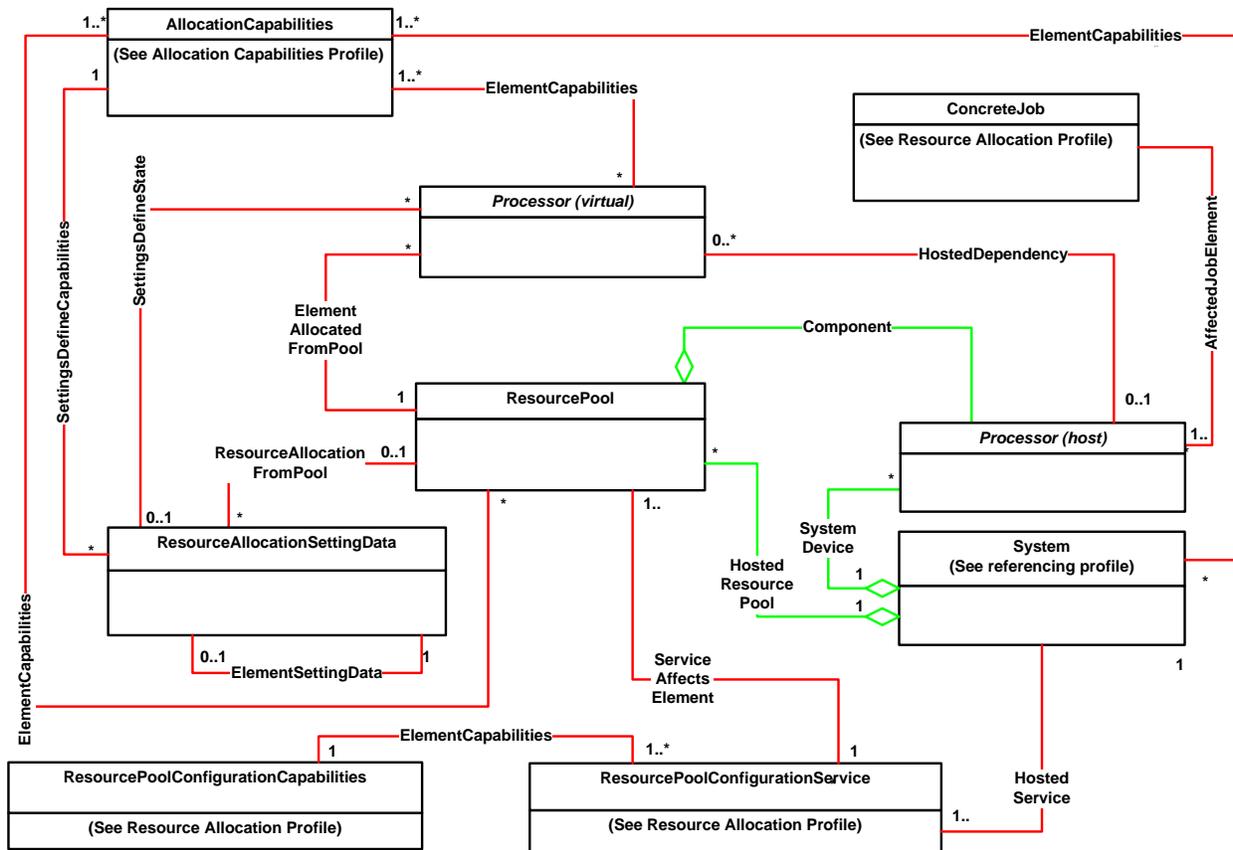
287 **6.1 General**

288 In computer virtualization systems, virtual computer systems are composed of component virtual
 289 resources. This profile specifies the allocation and management of host processor resources in support of
 290 virtual processors. From an operating system or application viewpoint virtual processors are functionally
 291 equivalent to "physical" processors.

292 This profile applies the resource virtualization pattern defined in [DSP1041](#) and the allocation capabilities
 293 pattern defined in [DSP1043](#) to enable the management of processor resources that are allocated to
 294 virtual systems. This profile defines additional CIM elements and constraints beyond those defined in the
 295 specialized profiles. Optionally implementations may implement [DSP1022](#) to represent host processors.

296 **6.2 Processor resource virtualization class schema**

297 Figure 1 represents the class schema of this profile. It outlines the elements that are adapted by this
 298 profile. For simplicity, the prefix CIM_ has been removed from the name of the classes.



299

300

Figure 1 – Processor Resource Virtualization Profile: Class Diagram

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

- 302 • The CIM_ResourcePool class models resource pools for processor resources. Host processor
303 resources are allocated from their resource pool and used to create the virtual processors for
304 virtual systems
- 305 • The CIM_Component association models the relationship between processor resource pools
306 and host processors as components of the resource pools
- 307 • The CIM_ElementAllocatedFromPool association models hierarchies of processor resource
308 pools and modeling the relationship of processor resource pools and the virtual processors
309 allocated from those
- 310 • The CIM_HostedResourcePool association models the hosting dependency between a
311 processor resource pool and its host system. A host system supports at least one processor
312 resource pool
- 313 • The CIM_Processor class models the following kinds of processors:
 - 314 – processors as a device in the scope of a system, as modeled by the CIM_SystemDevice
315 association
 - 316 – processors as a result of a processor resource allocation from a resource pool, as modeled
317 by the CIM_ElementAllocatedFromPool association
 - 318 – processors as a component within processor resource pools, as modeled through the
319 CIM_Component association

- 320 • The CIM_ResourceAllocationSettingData class models processor resource allocations or
321 processor resource allocation requests
- 322 • The CIM_AllocationCapabilities class and the CIM_ElementCapabilities association model
323 – the processor resource allocation capabilities of host systems
324 – the processor resource allocation capabilities of processor resource pools
325 – the mutability of existing processor resource allocations
- 326 • The CIM_SettingsDefineCapabilities association models the relation between processor
327 resource allocation capabilities and the settings that define these capabilities
- 328 • The CIM_ResourcePoolConfigurationService models configuration services for processor
329 resource pools and the CIM_ResourcePoolConfigurationCapabilities class modeling their
330 capabilities
- 331 • The CIM_ConcreteJob class and the CIM_AffectedJobElement association models
332 asynchronous management tasks initiated through memory resource pool configuration
333 services

334 In general, any mention of a class in this document means the class itself or its subclasses. For example,
335 a statement such as "an instance of the CIM_Processor class" implies an instance of the CIM_Processor
336 class or a subclass of the CIM_Processor class.

337 **6.3 Resource pools**

338 This profile applies the concept of resource pools defined in the [DSP1041](#) to the resource type 3
339 (Processor).

340 **6.3.1 General**

341 This profile uses processor resource pools as the focal point for processor allocations. A processor
342 resource pool represents an aggregate amount of processing capacity, and keeps track of the amount of
343 processor capacity that has been allocated to consumers such as virtual systems. A resource pool also
344 defines the scope in which relative weights are interpreted. A resource pool represents a part or all of the
345 aggregated processing power of a virtualization platform in an abstract sense, that is, it represents the
346 sum of the processing power of the host resources aggregated into the resource pool and is expressed in
347 allocation units such as MHz, percentage, or count of processors.

348 Note that the resource type of a resource pool governs the type of the resources that are allocated from
349 the resource pool. Opposed to that the resource type of the resources that are aggregated by the
350 resource pool may differ from the resource type of the pool. For example, a resource pool with a resource
351 type of 3 (Processor) supports the allocation of virtual processors. However, the resources that are
352 aggregated by that resource pool may be of a different type; for example, that resource pool might
353 aggregate processor fragments, or it might simply represent a certain amount of processing power
354 without representing individual processors.

355 **6.3.2 Representation of host resources**

356 A processor resource pool represents an aggregated amount of processing power provided by host
357 resources that enables the allocation of virtual processors. However the explicit representation of the host
358 resources aggregated by a resource pool is optional: In some cases implementations may explicitly
359 represent the host resources such as for example host processors. In other cases implementations may
360 choose not to explicitly represent the host resources aggregated by a resource pool, that is, an
361 implementation may choose to model a resource pool as the sole model element that represents host
362 processing capacity for the support of (allocated) virtual processors, but not detail the host resources that
363 provide that processing capacity.

364 The processor resources of a host system can be represented by a single processor resource pool, with
365 capacity equal to the processing capacity of the host computer system. The processor resources of a
366 host computer system may be represented by multiple resource pools, providing more flexible control
367 over the allocation of processor resources to virtual systems.

368 **6.3.3 Hierarchies of processor resource pools**

369 This profile applies the concept of resource pool hierarchies defined in [DSP1041](#) to the processor
370 resource type; see the "Hierarchies of Resource Pools" subclause in [DSP1041](#).

371 **6.4 Resource allocation**

372 This profile applies the concept of device resource allocation defined in [DSP1041](#) to the processor
373 resource type; see the "Device Resource Allocation" subclause in [DSP1041](#).

374 **6.4.1 Processor resource allocation request**

375 The processor requirements of a virtual system are defined as part of the "defined" virtual system
376 configuration; see [DSP1057](#) for a definition of the "defined" virtual system configuration. The "defined"
377 virtual system configuration contains processor resource allocation requests represented as RASD
378 instances.

379 **6.4.2 Processor Resource allocation**

380 As a virtual system is activated (or instantiated), one or more virtual processors need to be allocated as
381 requested by processor resource allocation requests in the virtual system definition. Processor resource
382 allocations are represented as RASD instances in the "state" virtual system configuration. The number of
383 discrete processors exposed to the virtual system is specified by the value of the VirtualQuantity property.

384 The central properties describing a processor resource allocation request or a processor resource
385 allocation are Reservation, Limit, and Weight.

386 The values of both the Reservation and Limit properties are specified in allocation units as expressed by
387 the value of the AllocationUnits property. Possible allocation units are a frequency (i.e., "Hertz" or "MHz"),
388 a percentage (with respect to some base value such as the sum of all available processing power), or a
389 count (expressing a number of processors or processor fractions to be allocated).

390 The value of the Reservation property specifies a lower bound on the quantity of host processor
391 resources available for the virtual computer system that are guaranteed to be available for use. If a virtual
392 computer system does not consume its full allocation of reserved resources, the host system may allow
393 its unused portion to be utilized by other virtual computer systems. The value of the Limit property
394 specifies a limit or upper bound on the quantity of host processor resources that may be consumed by the
395 virtual computer system. A limit is an artificial cap that may not be exceeded, even if otherwise-idle host
396 processor resources are available.

397 The value of the Weight property specifies the relative importance of the set of allocated virtual
398 processors, and is expressed in abstract numeric units. A virtual system is entitled to consume host
399 processor resources at a rate that is directly proportional to the value of the Weight property.

400 **6.4.3 Virtual processors**

401 A virtual processor is the instantiation of allocated processor resources that is exposed to a virtual system
402 through a logical processor device; it is the result of the processor resource allocation based on a
403 processor resource allocation request. A virtual processor may be realized using techniques such as time
404 sharing, but may also be a host processor that is directly passed through to the virtual system.

405 A virtual processor is represented by a CIM_Processor instance that is part of the virtual system
406 representation.

407 **6.4.4 Dedicated processors**

408 A dedicated host processor is a processor owned by the host system that is exclusively reserved for
409 support of a virtual processor of a particular virtual system.

410 **6.4.5 Consumption of host processing power**

411 A processor resource allocation request references the processor resource pool to be used by specifying
412 the value of the PoolID property. Host processor resources are allocated from the identified processor
413 resource pool during processor resource allocation.

414 **6.4.5.1 Statically controlled processor resource consumption**

415 With statically controlled processor resource consumption, a particular processor resource allocation
416 request is granted only if the amount of host processing resource available in the addressed processor
417 resource pool is at least as large as the amount requested. This approach is called admission control.
418 Each successfully allocated processor resource reduces the amount of processing resources available
419 from the pool, respectively. In the CIM representation of the processor resource pool, the amount of
420 processing power assigned to consumers from the pool is visible through the value of the Reserved
421 property. With admission control the processing capacity represented by a particular resource pool
422 remains larger than the sum of all reservations out of that pool. Admission control checks are typically
423 performed for the following operations: creating a resource pool, powering on a virtual system, and
424 modifying the resource allocation settings for either a resource pool or a running virtual system.

425 **6.4.5.2 Dynamically controlled processor resource consumption**

426 With dynamically controlled processor resource consumption, the amount of host processing resources
427 allocated in support of a virtual processor is not a constant value but varies significantly over time,
428 depending on factors like the processing requirements of the software executed within the virtual system
429 or the processing power consumption of other virtual machines.

430 Further, with dynamically controlled processor resource consumption, the amount of processing power
431 managed through a processor resource pool conceptually may be considered as unlimited, such that no
432 admission control is performed at the time virtual processors are initially allocated (usually at virtual
433 system activation time). Of course, this approach may result in an over commitment situation where the
434 host system experiences processing power shortages at a later point in time, such that ultimately the host
435 system is no longer able to support the sum of processing power requests of all hosted virtual systems.

436 **7 Implementation**

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

439 **7.1 Allocation units**

440 This subclause details requirements for the unit of measurement that applies to the specification of
441 processor resource allocations.

442 **7.1.1 General**

443 Processor resource allocations and processor resource allocation requests shall be expressed through
444 [DSP0004](#) programmatic units using one these base units: "hertz", "percent" or "count" .

445 7.1.2 Use of the base unit "hertz"

446 If the base unit of "hertz" is used, the following provisions apply:

- 447 • CIM_Processor class
 - 448 – MaxClockSpeed property: is expressed in MHz as defined in the CIM Schema
 - 449 – CurrentClockSpeed property: is expressed in MHz as defined in the CIM Schema
- 450 • CIM_ResourceAllocationSettingData class
 - 451 – AllocationUnits property: Value shall use a base unit of "hertz".
 - 452 – Reservation property: Value shall be expressed in the unit expressed by the value of the
 - 453 AllocationUnits property.
 - 454 – Limit property: Value shall be expressed in the unit expressed by the value of the
 - 455 AllocationUnits property.
- 456 • CIM_ResourcePool class
 - 457 – AllocationUnits property: Value shall use a base unit of "hertz".
 - 458 – Capacity property: Value shall be expressed in the unit expressed by the value of the
 - 459 AllocationUnits property.
 - 460 – Reserved property: Value shall be expressed in the unit expressed by the value of the
 - 461 AllocationUnits property.

462 7.1.3 Use of the base unit "percent"

463 If the base unit of "percent" is used, the following provisions apply:

- 464 • CIM_Processor class
 - 465 – MaxClockSpeed: Value is expressed in MHz as defined in the CIM Schema
 - 466 – CurrentClockSpeed: Value is expressed in MHz as defined in the CIM Schema
- 467 • CIM_ResourceAllocationSettingData class
 - 468 – AllocationUnits property: Value shall be "percent"
 - 469 – Reservation property: Value shall be expressed in percent, stating a minimum percentage
 - 470 of the processing power as requested from the resource pool identified by the RASD
 - 471 instance.
 - 472 – Limit property: Value shall be expressed in percent, stating a maximum percentage of the
 - 473 processing power to be provided by the resource pool identified by the RASD instance..
- 474 • CIM_ResourcePool class
 - 475 – AllocationUnits property: Value shall be "percent"
 - 476 – Capacity property: Value shall be expressed in the unit expressed by the value of the
 - 477 AllocationUnits property.
 - 478 – Reserved property: Value shall be expressed in percent, stating the reserved percentage
 - 479 of processing power presently allocated from the pool.

480 7.1.4 Use of the base unit "count"

481 If the base unit of "count" is used the following provisions apply:

- 482 • CIM_Processor class
 - 483 – MaxClockSpeed: Value is expressed in MHz as defined in the CIM Schema

- 484 – CurrentClockSpeed: Value is expressed in MHz as defined in the CIM Schema
- 485 • CIM_ResourceAllocationSettingData class
- 486 – AllocationUnits property: Value shall use a base unit of "count", the counted items shall be
487 processors. For example, a unit of a tenth of a processor can be expressed using the
488 value "count*0.1".
- 489 – Reservation property: Value shall be expressed in the unit expressed by the value of the
490 AllocationUnits property.
- 491 – Limit property: Value shall be expressed in the unit expressed by the value of the
492 AllocationUnits property.
- 493 • CIM_ResourcePool class
- 494 – AllocationUnits property: Value shall be use a base unit of "count" the counted items shall
495 be processors. For example, a unit of a tenth of a processor can be expressed using the
496 value "count*0.1".
- 497 – Capacity property: Value shall be expressed in the unit expressed by the value of the
498 AllocationUnits property.
- 499 – Reserved property: Value shall be expressed in the unit expressed by the value of the
500 AllocationUnits property.

501 **7.2 Host resources**

502 The implementation of the representation of host processor resources is optional.

503 If the representation of host processors is implemented, the provisions in this subclause apply.

504 Each host processor shall be represented by exactly one CIM_Processor instance that is associated with
505 the CIM_System instance that represents the host system through an instance of the CIM_SystemDevice
506 association.

507 Implementations may implement [DSP1022](#) for host processors.

508 **7.3 Resource pools**

509 This subclause adapts the CIM_ResourcePool class for the representation of processor resource pools.

510 **7.3.1 ResourceType property**

511 The value of the ResourceType property shall be 3 (Processor).

512 **7.3.2 ResourceSubType property**

513 The implementation of the ResourceSubType property is optional.

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

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

519 **7.3.3 Primordial property**

520 The value of the Primordial property shall be set to TRUE for any CIM_ResourcePool instance that
521 represents a primordial processor resource pool. For other CIM_ResourcePool instances that represent
522 processor resource pools, the value of the Primordial property shall be set to FALSE.

523 **7.3.4 PoolID property**

524 The value of the PoolID property shall be set such that it enables unique identification of the
525 CIM_ResourcePool instance within the scoping host system.

526 **7.3.5 Reserved property**

527 The implementation of the Reserved property is optional.

528 If the Reserved property is implemented, its value shall reflect the amount of host processing resource
529 that is actually reserved from the resource pool, in units as expressed by the value of the AllocationUnits
530 property (see 7.3.7) .

531 **7.3.6 Capacity property**

532 The implementation of the Capacity property is conditional.

533 Condition: The Capacity property shall be implemented if the representation of the aggregation of host
534 resources is implemented (see 7.4).

535 If the Capacity property is implemented, its value shall reflect the maximum amount of processing power
536 that can be allocated from the resource pool, in units as expressed by the value of the AllocationUnits
537 property (see 7.3.7). If the CIM_ResourcePool instance represents a processor resource pool with
538 unlimited capacity, the value of the Capacity property shall be set to the largest value supported by the
539 uint64 datatype.

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

542 **7.3.7 AllocationUnits property**

543 The value of the AllocationUnits property shall be expressed through [DSP0004](#) programmatic units using
544 one these base units: "hertz", "percent" or "count" .

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

547 **7.3.8 Instance requirements**

548 Each processor resource pool shall be represented by a CIM_ResourcePool instance; the provisions of
549 10.12 and 7.3 apply. The CIM_ResourcePool instance shall be associated with the CIM_System instance
550 representing the host system through an instance of the CIM_HostedResourcePool association (see
551 [DSP1041](#)).

552 **7.4 Aggregation of host resources**

553 The implementation of the representation of the aggregation of host processor(s) into a processor
554 resource pool is optional.

555 If the representation of the aggregation of host processors is implemented, it may be supported for all or
556 for only for some processor resource pools. If the aggregation of host processors is supported for a
557 particular resource pool, any instance of the CIM_Processor class representing a host processor that
558 contributes processing resources into that resource pool shall be associated to the CIM_ResourcePool
559 instance representing the resource pool through an instance of the subclass of CIM_Component
560 association; the provisions of 10.1 apply.

561 7.5 Processor resource pool hierarchies

562 The implementation of the representation of processor resource pool hierarchies is optional. If
 563 implemented, any concrete processor resource pool shall be represented through a CIM_ResourcePool
 564 instance, where all of the following conditions shall be met:

- 565 • The value of the Primordial property shall be FALSE.
- 566 • The instance shall be associated through an instance of CIM_ElementAllocatedFromPool
 567 association to the CIM_ResourcePool instance that represents its parent processor resource
 568 pool.
- 569 • The instance shall be associated through an instance of the CIM_ElementSettingData
 570 association to the RASD instance that represents the amount of processing power allocated
 571 from the parent resource pool.

572 7.6 Default processor resource pool

573 The implementation of designating a default processor resource pool is optional. If implemented, all of
 574 the following conditions apply:

- 575 • The default processor resource pool shall be represented by a CIM_ResourcePool instance;
 576 see 7.8.3.2
- 577 • That instance shall be associated to the CIM_AllocationCapabilities instance that represents the
 578 pools default allocation capabilities as specified in 7.8.4.3 .
- 579 • The same CIM_AllocationCapabilities instance shall also represent the systems default
 580 allocation capabilities as specified in 7.8.3.2.

581 7.7 Processor resource pool management

582 The implementation of processor resource pool management is optional.

583 If implemented, the specifications of [DSP1041](#), clause 7.4 "Resource Pool Management" apply; this
 584 profile does not specify specializations or extensions of resource pool management beyond those defined
 585 by [DSP1041](#) .

586 7.8 Processor resource allocation

587 This subclause details requirements for the representation of resource allocation information through
 588 CIM_ResourceAllocationSettingData (RASD) instances.

589 7.8.1 General

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

592 Implementations of this profile shall implement the *virtual resource allocation* pattern as defined in
 593 [DSP1041](#), 7.2.

594 7.8.2 Flavors of allocation data

595 Various flavors of allocation data are defined:

- 596 • Processor resource allocation requests; see 6.4.1 .
- 597 • Processor resource allocations; see 6.4.2 .
- 598 • Settings that define the capabilities or mutability of managed resources. [DSP1043](#) specifies a
 599 capabilities model that conveys information about the capabilities and the mutability of managed
 600 resources in terms of RASD instances.

- 601 • Parameters in operations that define or modify any of the representations listed above.
 602 [DSP1042](#) that specifies methods for the definition and modification of virtual resources. These
 603 methods use RASD instances for the parameterization of resource-allocation-specific
 604 properties.

605 Table 2 lists acronyms that are used in subclauses of 7.8 in order to designate RASD instances that
 606 represent various flavors of processor resource allocation data.

607 **Table 2 – Acronyms for RASD adapted for the representation of various flavors of allocation data**

Acronym	Flavor
Q_RASD	RASD adapted for the representation of processor resource allocation requests
R_RASD	RASD adapted for the representation of processor resource allocations
C_RASD	RASD adapted for the representation of settings that define capabilities of systems or processor resource pools, or that define the mutability of processor resource allocations or processor resource allocation requests
D_RASD	RASD adapted for the representation of new processor resource allocation requests in method parameter values
M_RASD	RASD adapted for the representation of modified processor resource allocations or processor resource allocation request in method parameter values

608 Subclauses of 7.8 detail implementation requirements for property values in RASD instances. In some
 609 cases requirements only apply to a subset of the flavors listed in Table 2; this is marked in the text
 610 through the use of respective acronyms.

611 **7.8.3 CIM_ResourceAllocationSettingData class**

612 This subclause defines rules for the values of properties in instances of the
 613 CIM_ResourceAllocationSettingData (RASD) class representing processor resource allocation
 614 information representing the various flavors of processor resource allocation information defined in Table
 615 2.

616 **7.8.3.1 ResourceType property**

617 The value of the ResourceType property in RASD instances representing processor resource allocation
 618 information shall be set to 3 (Processor) for processor resource allocation data.

619 **7.8.3.2 PoolID property**

620 The value of the PoolID property shall designate the processor resource pool. A NULL value shall
 621 indicate the use of the host system's default processor resource pool.

622 **7.8.3.3 ConsumerVisibility property**

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

- 625 • A value of 2 (Passed-Through) shall denote that the virtual processor is based on a
 626 passed-through host processor .
- 627 • A value of 3 (Virtualized) shall denote that processor is virtualized.
- 628 • In the cases of { Q_RASD | D_RASD | M_RASD), a value of 0 (Unknown) shall indicate that
 629 the processor resource allocation request does not predefine which type of processor shall be
 630 allocated.

631 Other values shall not be used.

632 **7.8.3.4 HostResource[] array property**

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

634 Condition: The value 2 (Passed-Through) is supported for the value of the ConsumerVisibility property,
635 or any of the values 3 (Dedicated), 4 (Soft Affinity) or 5 (Hard Affinity) is supported for the
636 MappingBehavior property.

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

638 In the cases of { Q_RASD | C_RASD | D_RASD | M_RASD } the value of the HostResource[] array
639 property shall refer to (the representation of) one or more host resources that are configured to contribute
640 processing power for the processor resource allocation.

641 In the case of R_RASD the value of the HostResource[] array property shall refer to (the representation
642 of) the host resource(s) that contribute processing power for the processor resource allocation.

643 Elements of the value of the HostResource[] array property shall refer to instances of CIM classes, using
644 the WbemUri format as specified by [DSP0207](#).

645 **7.8.3.5 AllocationUnits property**

646 The value of the AllocationUnits property shall be expressed in one of the following programmatic units:
647 "hertz", "percent" or "count" or a multiple of the units expressed through a regular expression, as defined
648 in [DSP0004](#) programmatic units.

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

651 **7.8.3.6 VirtualQuantity property**

652 The value of the VirtualQuantity property shall denote the number of virtual processors available to a
653 virtual system.

654 NOTE The value of the VirtualQuantity property is a count; it is not expressed in allocation units. The units used
655 for VirtualQuantity may be expressed in VirtualQuantityUnits (see 7.8.3.11).

656 **7.8.3.7 Reservation property**

657 The implementation of the Reservation property is optional.

658 If the Reservation property is implemented, the value of the Reservation property shall denote the
659 minimum amount of host processing resources reserved for the use of a virtual system, expressed in
660 allocation units.

661 **7.8.3.8 Limit property**

662 The implementation of the Limit property is optional.

663 If the Limit property is implemented, the value of the Limit property shall denote the maximum amount (or
664 limit) of host processing power available to a virtual system, expressed in allocation units.

665 **7.8.3.9 Weight property**

666 The implementation of the Weight property is optional.

667 If the Weight property is implemented, its value shall denote the relative priority of a processor resource
668 allocation in relation to other processor resource allocations from the same resource pool.

669 **7.8.3.10 MappingBehavior property**

670 The implementation of the MappingBehavior property is optional.

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

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

- 674 • A value of 2 (Dedicated) shall indicate that the represented processor resource allocation is
675 provided by host processor resources as referenced by the value of the HostResource[] array
676 property that are exclusively dedicated to the virtual system.
- 677 • A value of 3 (Soft Affinity) or 4 (Hard Affinity) shall indicate that the represented processor
678 resource allocation is provided using host processor resource as referenced by the value of the
679 HostResource[] array property.
- 680 • Other values shall not be used.

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

- 682 • The special value NULL or a value of 0 (Unknown) shall indicate that the processor resource
683 allocation request does not require specific host resources.
- 684 • A value of 2 (Dedicated) shall indicate that the processor resource allocation request shall be
685 provided by exclusively dedicated host processor resources as specified through the value of
686 the HostResource[] array property.
- 687 • A value of 3 (Soft Affinity) shall indicate that the processor resource allocation request shall
688 preferably be provided by host processor resources as specified through the value of the
689 HostResource[] array property, but that other resources may be used if the requested
690 resources are not available.
- 691 • A value of 4 (Hard Affinity) shall indicate that the processor resource allocation request shall
692 preferably be provided by host processor resources as specified through the value of the
693 HostResource[] array property and that other resources shall not be used if the requested
694 resources are not available.
- 695 • Other values shall not be used.

696 **7.8.3.11 VirtualQuantityUnits property**

697 VirtualQuantityUnits is an optional property that may be used to denote the units of the virtual device
698 being allocated from the resource pool. The value of VirtualQuantityUnits shall be "count".

699 **7.8.3.12 ResourceSubType property**

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

704 **7.8.4 Instance requirements**

705 This subclause details processor resource allocation related instance requirements.

706 **7.8.4.1 Representation of resource allocation requests**

707 Each processor resource allocation request shall be represented by a Q_RASD instance; the provisions
708 of 10.11 apply.

709 **7.8.4.2 Representation of resource allocations**

710 Each processor resource allocation shall be represented by a R_RASD instance; the provisions of 10.11
711 apply.

712 The R_RASD instance shall be associated to the Q_RASD instance representing the corresponding
713 resource allocation request (see 7.8.4.1) through an instance of the CIM_ElementSettingData
714 association; the provisions of 10.10 apply.

715 The R_RASD instance shall be associated to the CIM_ResourcePool instance providing resources for the
716 allocation (see 7.3.8) through an instance of the CIM_ResourceAllocationFromPool association; the
717 provisions of 10.4 apply.

718 Implementations may represent a resource allocation request and the corresponding resource allocation
719 by one RASD instance; in this case the association requirements of this subclause apply correspondingly.
720 Note that association instances that refer to the RA_SASD instance are only existent while the resource
721 is allocated.

722 **7.8.4.3 Representation of resource allocation capabilities**

723 The allocation capabilities of a system or a resource pool shall be represented by a
724 CIM_AllocationCapabilities instance that is associated to the CIM_System instance representing the
725 system or to the CIM_ResourcePool instance representing the resource pool through an instance of the
726 CIM_ElementCapabilities association; see [DSP1043](#).

727 The settings that define the allocation capabilities of a processor resource pool shall be represented by
728 C_RASD instances; the provisions of 10.11 apply.

729 The processor allocation capabilities of a host system shall be a superset of the processor allocation
730 capabilities of all processor resource pools that are hosted by the host system.

731 **7.8.4.4 Representation of resource allocation mutability**

732 The mutability of a resource allocation or resource allocation request shall be represented by a
733 CIM_AllocationCapabilities instance that is associated to the RASD instance representing the resource
734 allocation or resource allocation request through an instance of the CIM_ElementCapabilities association;
735 see [DSP1043](#).

736 The settings that define the allocation capabilities of a processor resource pool shall be represented by
737 C_RASD instances; the provisions of 10.11 apply.

738 **7.9 Virtual processor**

739 A virtual processor shall be represented by exactly one CIM_Processor instance; the provisions of 10.8
740 apply. That instance shall be associated to all of the following instances:

- 741 • the CIM_ComputerSystem instance that represents the virtual system through an instance of
742 the CIM_SystemDevice association; the provisions of 10.15 apply.
- 743 • the RASD instance that represents processor resource allocation through an instance of the
744 CIM_SettingsDefineState association; the provisions of 10.13 apply.
- 745 • the CIM_ResourcePool instance that represents the processor resource pool providing the
746 resource allocation through an instance of the CIM_ElementAllocatedFromPool association; the
747 provisions of 10.2 apply.

748 Implementations may implement [DSP1022](#) for virtual processors.

749 **8 Methods**

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

752 **8.1 Profile conventions for operations**

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

755 The default list of operations for all classes is:

- 756 • `GetInstance()`
- 757 • `EnumerateInstances()`
- 758 • `EnumerateInstanceNames()`

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

- 760 • `Associators()`
- 761 • `AssociatorNames()`
- 762 • `References()`
- 763 • `ReferenceNames()`

764 The implementation requirements for intrinsic operations and extrinsic methods of classes listed in clause
765 10, but not addressed by a separate subclause of this clause are specified by the "Methods" clauses of
766 respective base profiles, namely [DSP1041](#) and [DSP1043](#). These profiles are specialized by this profile,
767 and in these cases this profile does not add method specifications beyond those defined in its base
768 profiles.

769 **8.2 CIM_Processor for host processors**

770 All operations in the default list in 8.1 shall be implemented as defined in [DSP0200](#). Note that related
771 profiles may define additional requirements on operations for the profile class.

772 **8.3 CIM_Processor for virtual processor**

773 All operations in the default list in 8.1 shall be implemented as defined in [DSP0200](#). Note that related
774 profiles may define additional requirements on operations for the profile class.

775 **8.4 CIM_ReferencedProfile**

776 All operations in the default list in 8.1 shall be implemented as defined in [DSP0200](#). Note that related
777 profiles may define additional requirements on operations for the profile class.

778 **8.5 CIM_RegisteredProfile**

779 All operations in the default list in 8.1 shall be implemented as defined in [DSP0200](#). Note that related
780 profiles may define additional requirements on operations for the profile class.

781 **8.6 CIM_SystemDevice for host processors**

782 All operations in the default list in 8.1 shall be implemented as defined in [DSP0200](#). Note that related
783 profiles may define additional requirements on operations for the profile class.

784 **8.7 CIM_SystemDevice for virtual processors**

785 All operations in the default list in 8.1 shall be implemented as defined in [DSP0200](#). Note that related
786 profiles may define additional requirements on operations for the profile class.

787 **9 Use cases**

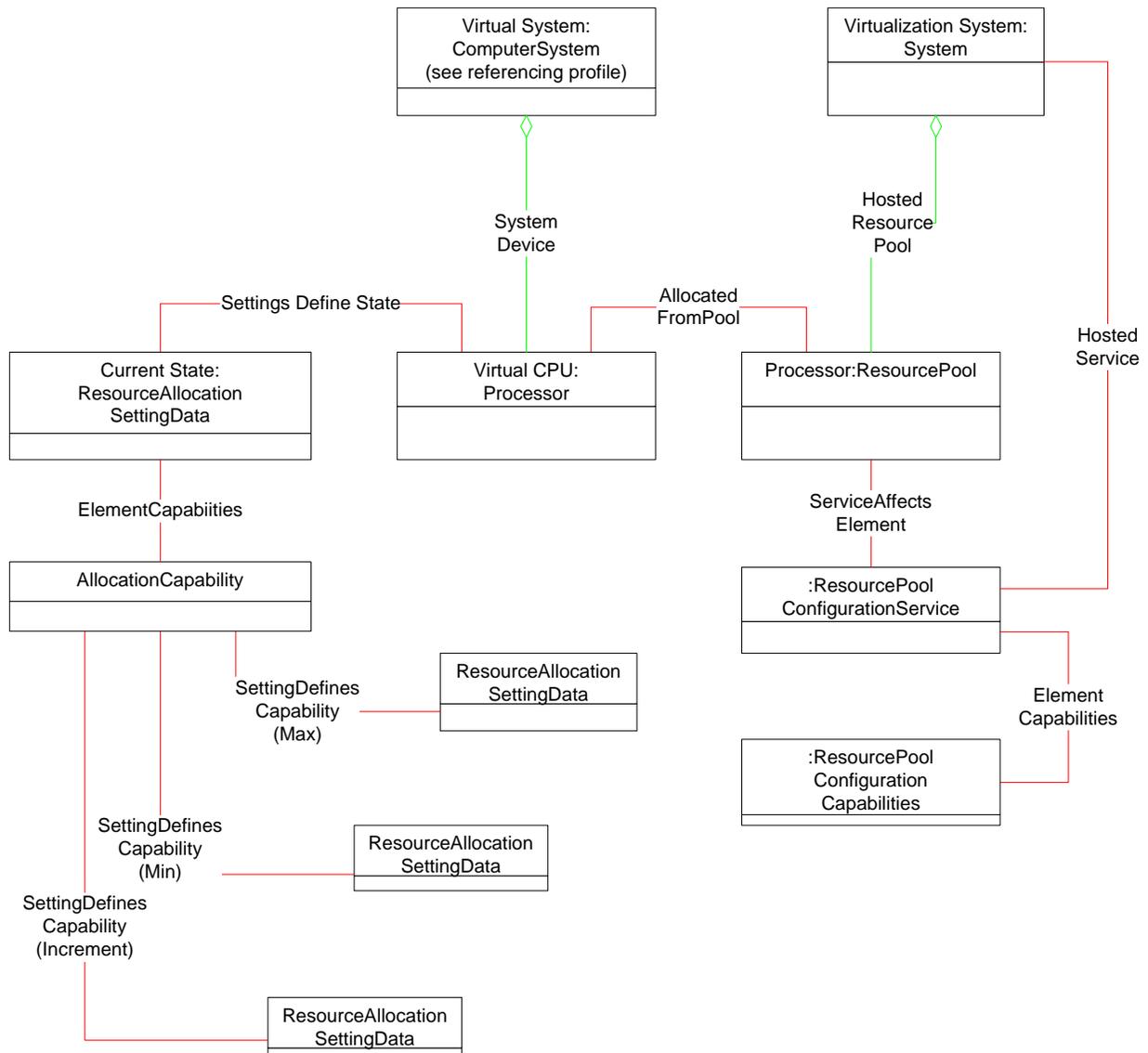
788 Clause 9 of [DSP1042](#) describes use cases virtual system definition, modification and destruction.

789 Clause 9 of [DSP1057](#) describes use cases for discovery of virtual systems, determination of the state and
790 properties of a virtual system and the defined virtual system. This clause is a pre-requisite to
791 understanding the following use cases.

792 Clause 9 of [DSP1059](#) covers a number of essential use cases and should be read.

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

795 Figure 2 is a general instance diagram showing the essential classes for processor resource allocation.



796

797

Figure 2 – Processor Resource Virtualization Profile: Instance diagram

798

There are different allocation units that may be used for the processor resource pool.

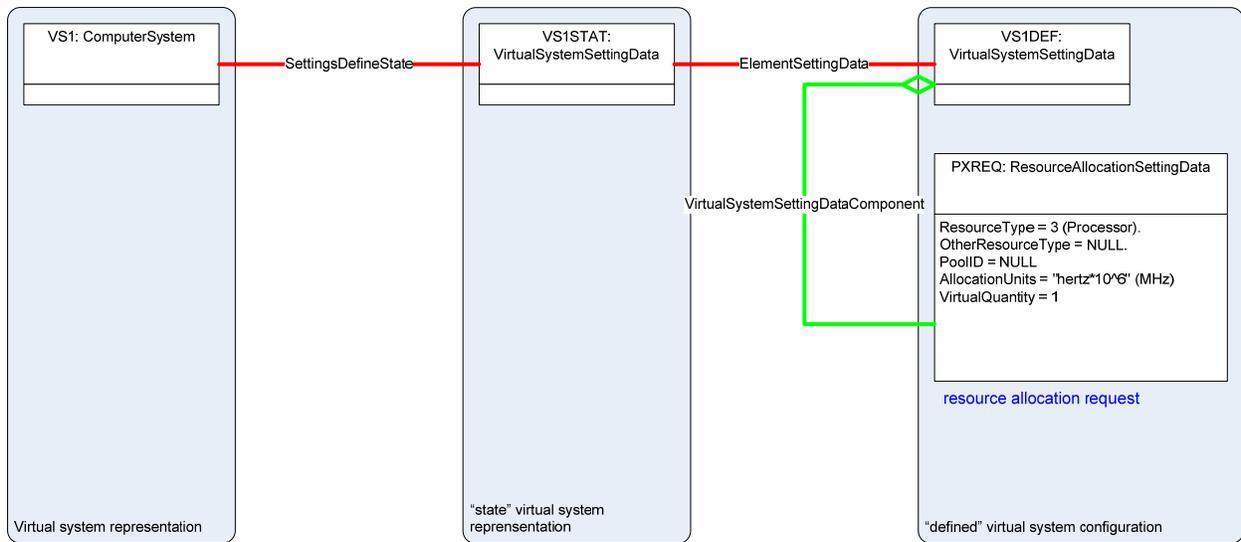
799

If instances of CIM_processor are instantiated to represent the physical processors aggregated into the primordial pool those instances should conform to [DSP1022](#).

800

801

A defined state for allocation of processor resources is shown in Figure 3.

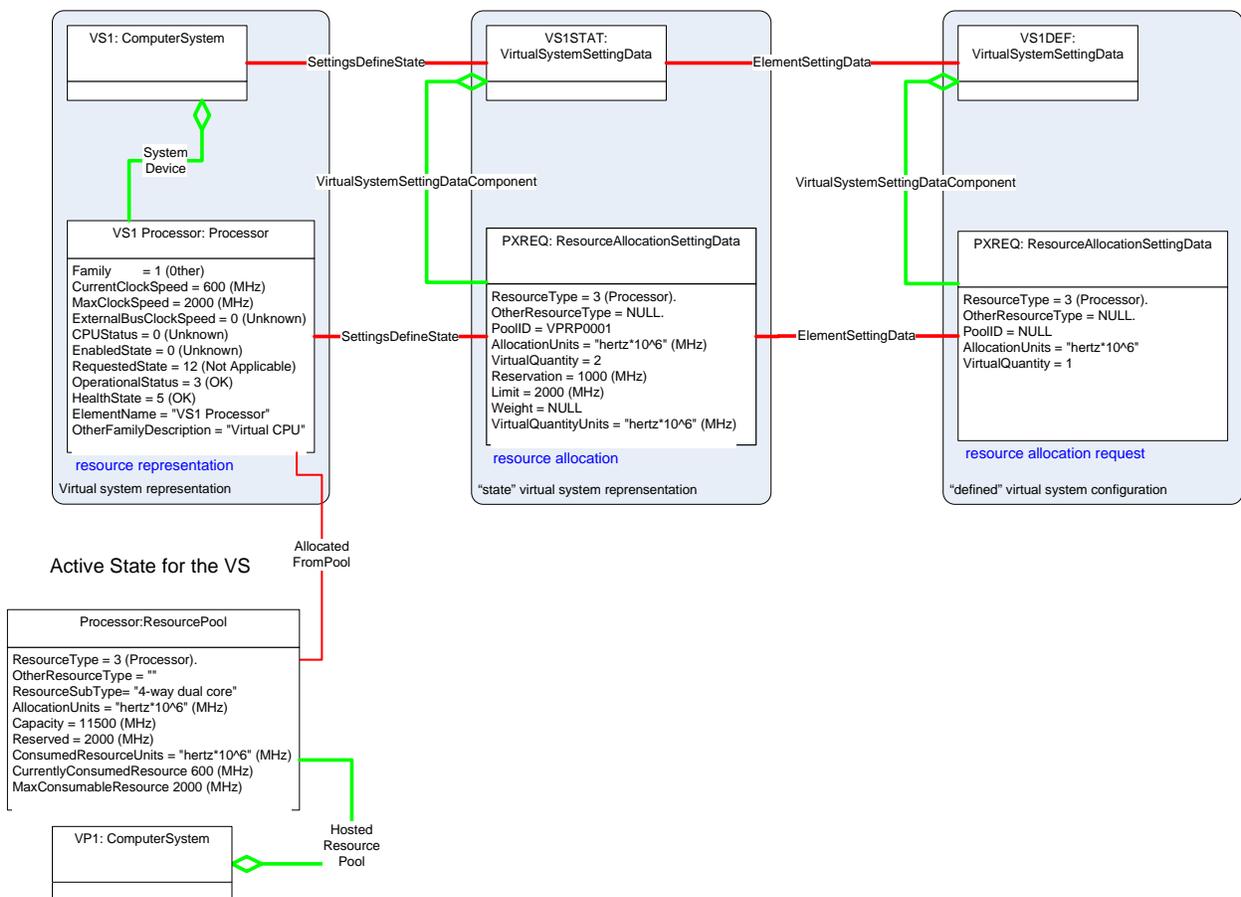


802

803

Figure 3 – Defined state

804 An active state for allocation of processor resources is shown in Figure 4.



805

806

Figure 4 – Active state

807 **9.1 Use case 1 – Increase the allocated processor capacity to a virtual machine**
808 **in MHz.**

809 Pre-conditions:

810 The virtual system is active.

811 The CIM_ComputerSystem instance representing the virtual system is known.

812 The CIM_ResourcePool instance representing the processor resource pool is known.

813 The CIM_AllocationUnits of the resource pool is “hertz*10⁶”.

814 The defined CIM_ResourceAllocationSettingData for the processor has property Reservation = 1000 (i.e
815 1GHz of processor capacity is allocated to the existing active virtual system) and has property Limit =
816 3000. The Reservation represents the minimum guaranteed resource available, even when system
817 overcommitted. The sum of the Reservations must be less than the capacity of the resource pool. The
818 Limit represents the maximum resource consumption, even when under committed.

819 Main:

820 The client changes the defined CIM_ResourceAllocationSettingData for the processor property to
821 Reservation = 2000.

822 The client invokes the CIM_VirtualManagementService.ModifyResourceSettings() method.

823 Check the return code value for success execution.

824 Post-conditions:

825 The minimum processor capacity required to be allocated for the support of the virtual processor is now 2
826 GHz, the maximum admissible processor capacity is now 3 GHz.

827 **9.2 Use case 2 – Increase the allocated processor capacity to a virtual machine**
828 **in percent.**

829 Pre-conditions:

830 The virtual system is active.

831 The CIM_ComputerSystem instance representing the virtual system is known.

832 The CIM_ResourcePool instance representing the processor resource pool is known.

833 The CIM_ResourceAllocationSettingData.AllocationUnits of the resource pool is “%”.

834 The defined CIM_ResourceAllocationSettingData for the processor has property Reservation = 10 (i.e ten
835 percent of processor capacity is allocated to the existing active virtual system) and has property Limit =
836 15.

837 Main:

838 The client changes the defined CIM_ResourceAllocationSettingData for the processor property
839 Reservation = 15.

840 The client invokes the CIM_VirtualSystemManagementService.ModifyResourceSettings() method.

841 Check the return code value for success execution.

842 Post-conditions:

843 The processor resource allocated to the virtual system is now 15 percent minimum and the maximum
 844 processor resource that the virtual system can consume is 15 percent.

845 **9.3 Use case 3 - Add a virtual processor to a virtual system**

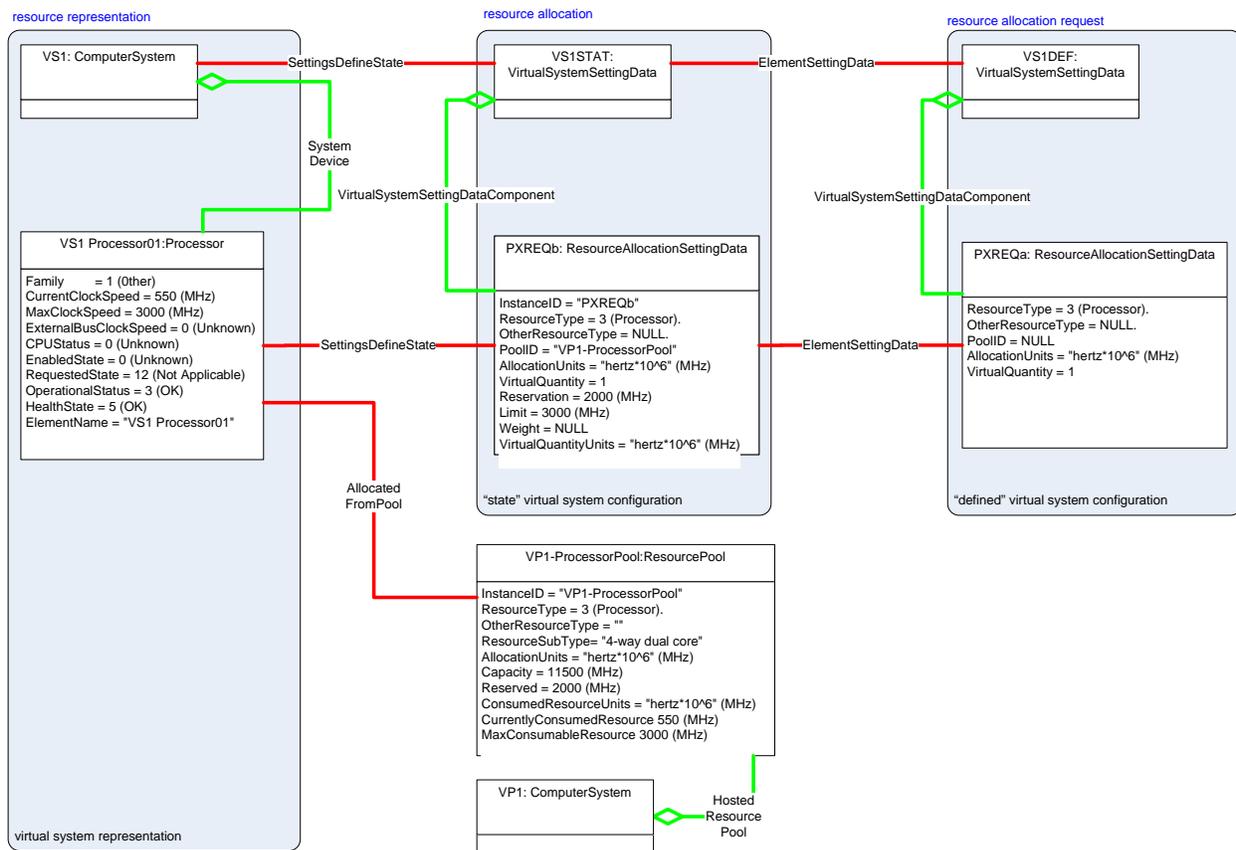
846 The virtual system is in-active.

847 The CIM_ComputerSystem instance representing the virtual system is known.

848 The CIM_ResourcePool instance representing the processor resource pool is known.

849 TheCIM_ResourceAllocationSettingData.AllocationUnits of the resource pool is "hertz*10^6".

850 Figure 5 illustrates the instance diagram before the
 851 CIM_VirtualSystemManagementService.ModifyResourceSettings() method is called.



852

853 **Figure 5 – CIM_ModifyResourceSettings – Before**

854 Figure 6 illustrates the resource allocation setting data transferred with the
 855 CIM_VirtualSystemManagementService.ModifyResourceSettings() method call to increase the number of
 856 virtual processors.

PXREQb: ResourceAllocationSettingData
InstanceID = PXREQb ResourceType = NULL OtherResourceType = NULL PoolID = NULL AllocationUnits = NULL VirtualQuantity = 2 Reservation = NULL Limit = NULL Weight = NULL VirtualQuantityUnits = NULL

857

858

Figure 6 – RASD to Modify Resources

859 Main:

860 Check the maximum value of the CIM_ResourceAllocationSettingData.VirtualQuantity to verify that a
861 virtual processor addition is allowed.

862 Check the increment value of the CIM_ResourceAllocationSettingData.VirtualQuantity to verify that a one
863 processor addition is allowed.

864 The client changes the CIM_ResourceAllocationSettingData.VirtualQuantity by one.

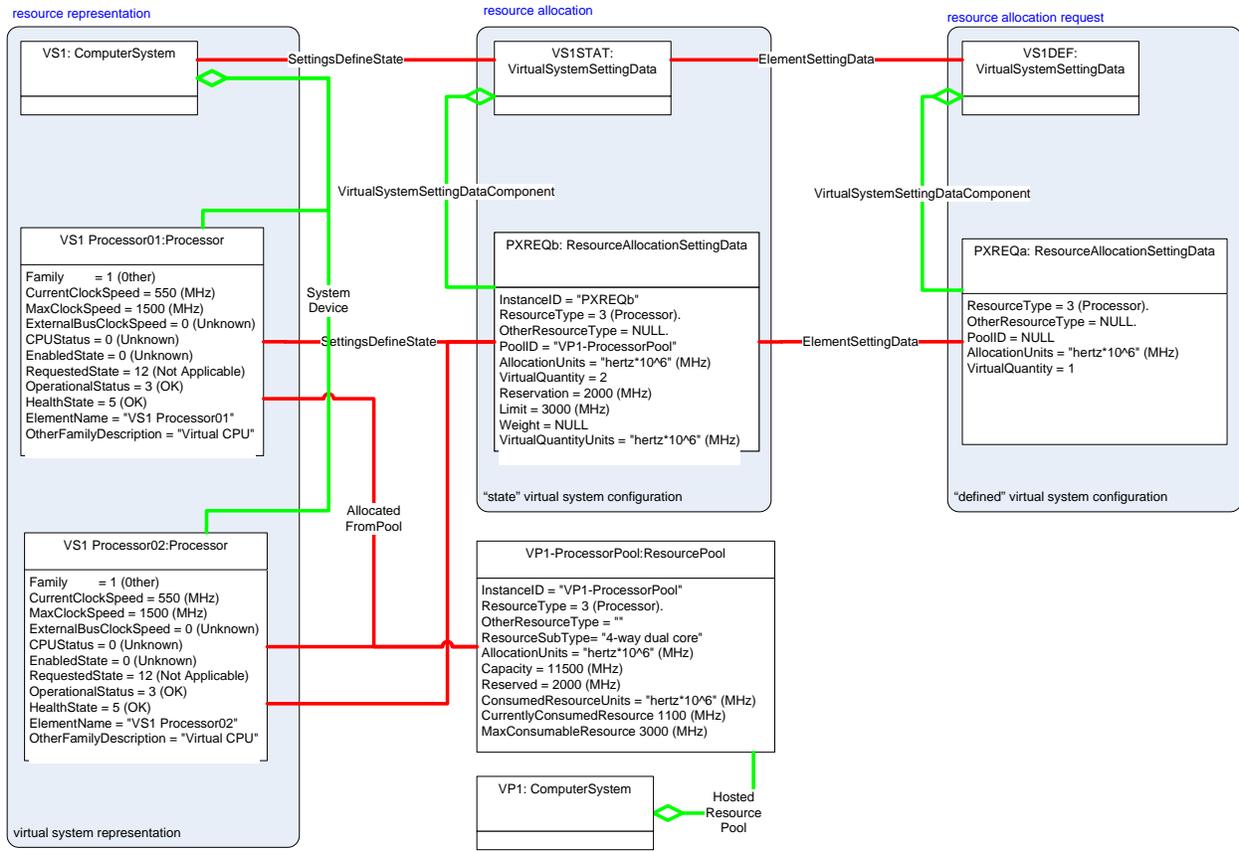
865 The client invokes the CIM_VirtualSystemManagement.Service.ModifyResourceSettings() method.

866 Check the return code value for success execution.

867 Post-conditions:

868 The virtual quantity of processors allocated to the virtual system is now one greater.

869 Figure 7 illustrates the instance diagram after the
870 CIM_VirtualSystemManagementService.ModifyResourceSettings() method is called.



871

872

Figure 7 – CIM_ModifyResourceSettings - After

873 **9.4 Use case 4 – Allocation of processor resource by weight**

874 Pre-conditions:

875 The virtual system is in-active.

876 The CIM_ComputerSystem instance representing the virtual system is known.

877 The CIM_ResourcePool instance representing the processor resource pool is known.

878 The CIM_AllocationUnits of the resource pool is “hertz*10^6”.

879 The defined CIM_ResourceAllocationSettingData for the processor has property Reservation = 0 (i.e no
 880 minimum level of processor capacity is allocated to the existing active virtual system) and has property
 881 Limit = none.

882 Main:

883 The client changes the defined CIM_ResourceAllocationSettingData for the processor property Weight =
 884 200.

885 The client invokes the CIM_VirtualSystemManagementService.ModifyResourceSettings() method.

886 Check the return code value for success execution.

887 The client activates the virtual system.

888 Post-conditions:

889 The processor resource allocated to the virtual system is now 200/ SUM (Weight for each active virtual
 890 system). For example, if there are three other virtual systems with their Weight properties set to 200, 300,
 891 100, then this virtual system is allocated $200/(200+300+100+200)$ or 25% of the processor capacity.

892 **9.5 Use case 5 – Allocation of an additional processor resource**

893 Pre-conditions:

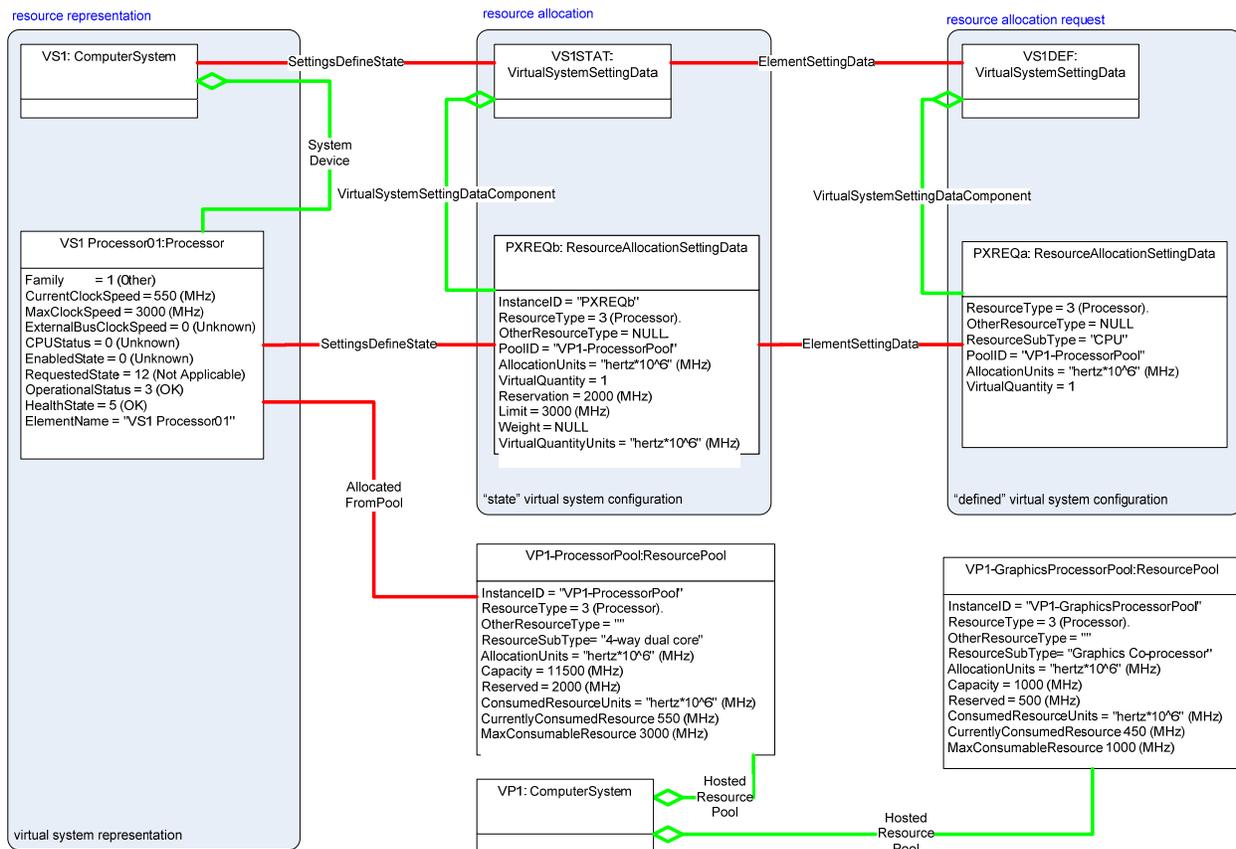
894 The virtual system is active.

895 The CIM_ComputerSystem instance representing the virtual system is known.

896 The CIM_ResourcePool instances representing the processor resource pools is known.

897 The CIM_AllocationUnits of the resource pools is "hertz 10^6".

898 Figure 7 illustrates the instance diagram before the
 899 CIM_VirtualSystemManagementService.AddResourceSettings() method is called.



900

901 **Figure 8 – CIM_AddResourceSettings - Before**

902 Figure 9 illustrates the resource allocation setting data transferred with the
 903 CIM_VirtualSystemManagementService.AddResourceSettings() method call to add an additional virtual
 904 processor to the virtual system.

GXREQa: ResourceAllocationSettingData
ResourceType = 3 (Processor). OtherResourceType = NULL. ResourceSubType = "GPU" PoolID = "VP1-GraphicsProcessorPool" AllocationUnits = "hertz*10^6" (MHz) VirtualQuantity = 1

905

906

Figure 9 – RASD to add processor

907 Main:

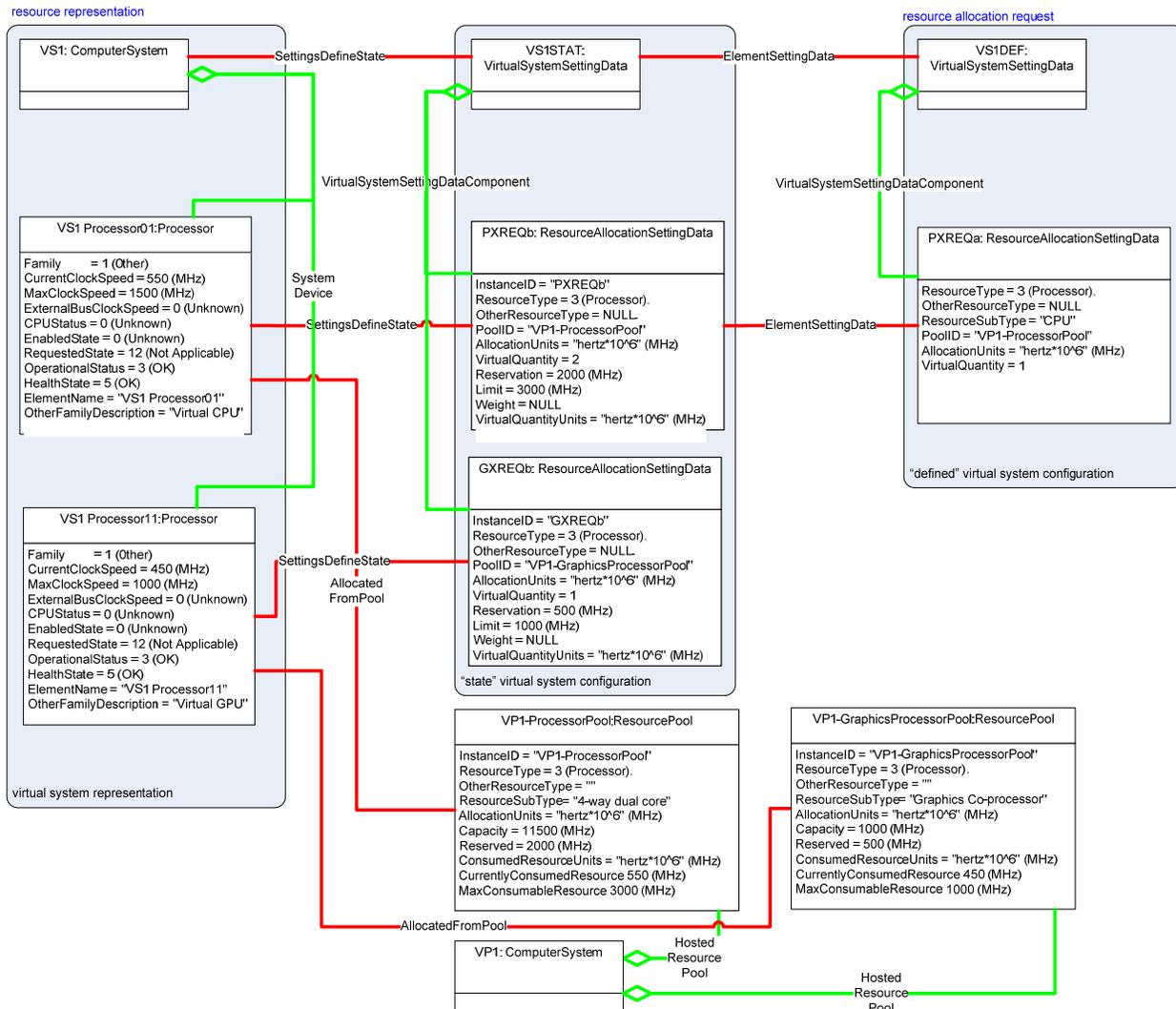
908 The client creates a defined CIM_ResourceAllocationSettingData specifying the PoolID for the resource
909 pool from which the additional processor is to be allocated.

910 The client invokes the CIM_VirtualSystemManagementService.AddResourceSettings() method.

911 Check the return code value for success execution.

912 Post-conditions:

913 The processor resources allocated to the virtual system is now two processors, one from the “VP1-
914 ProcessorPool” and one from the “VP1-GraphicsProcessorPool”. Figure 10 illustrates the instance
915 diagram after the CIM_VirtualSystemManagementService.AddResourceSettings() method is called.



916

917

Figure 10 – CIM_AddResourceSettings - After

918 **10 CIM elements**

919 Table 3 lists CIM elements that are adapted by this profile. Each CIM element shall be implemented as
 920 described in Table 3. The CIM Schema descriptions for any referenced element and its sub-elements
 921 apply.

922 Clauses 7 ("Implementation") and 8 ("Methods") may impose additional requirements on these elements.

923 **Table 3 – CIM Elements: Processor Resource Virtualization Profile**

Element Name	Requirement	Description
CIM_AffectedJobElement	Optional	See DSP1041 .
CIM_AllocationCapabilities for capabilities	Mandatory	See DSP1043 .
CIM_AllocationCapabilities for mutability	Optional	See DSP1043 .
CIM_Component for resource pool	Conditional	See 10.1 .
CIM_ConcreteJob	Optional	See DSP1041 .
CIM_ElementAllocatedFromPool for allocated virtual processors	Mandatory	See 10.2 .
CIM_ElementAllocatedFromPool for resource pool hierarchies	Conditional	See 10.3 .
CIM_ElementCapabilities for capabilities	Mandatory	See DSP1043 .
CIM_ElementCapabilities for mutability	Conditional	See DSP1043 .
CIM_ElementCapabilities for resource pools	Mandatory	See DSP1041 .
CIM_ElementSettingData for processor resource allocation	Mandatory	See 10.4 .
CIM_ElementSettingData for processor resource pools	Conditional	See 10.5 .
CIM_HostedDependency	Optional	See 10.6 .
CIM_HostedResourcePool	Mandatory	See DSP1041 .
CIM_HostedService	Mandatory	See DSP1041 .
CIM_Processor for host processors	Conditional	See 10.7 .
CIM_Processor for virtual processors	Mandatory	See 10.8 .
CIM_ReferencedProfile	Mandatory	See DSP1057 .
CIM_RegisteredProfile	Mandatory	See 10.9 .
CIM_ResourceAllocationFromPool	Optional	See 10.10 .
CIM_ResourceAllocationSettingData	Mandatory	See 10.11 .
CIM_ResourcePool	Mandatory	See 10.12 .
CIM_ResourcePoolConfigurationCapabilities	Mandatory	See DSP1041 .
CIM_ResourcePoolConfigurationService	Mandatory	See DSP1041 .
CIM_SettingsDefineState	Mandatory	See 10.13 .
CIM_ServiceAffectsElement	Mandatory	See DSP1041 .
CIM_SystemDevice for host processors	Conditional	See 10.14 .
CIM_SystemDevice for virtual processors	Mandatory	See 10.15 .
Indications		
None defined		

924 **10.1 CIM_Component for resource pool**

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

927 Condition: The representation of resource aggregation (see 7.4) is implemented.

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

932 Table 4 lists the requirements for elements of this association. These requirements are in addition to
 933 those specified in the CIM Schema and in [DSP1041](#).

934 **Table 4 – Association: CIM_Component for resource pool**

Elements	Requirement	Notes
GroupComponent	Mandatory	Key: Value shall reference the CIM_ResourcePool instance that represents the resource pool. Cardinality: 0..1
PartComponent	Mandatory	Key: Value shall reference the CIM_ManagedElement instance that represents a component of the resource pool. Cardinality: *

935 10.2 CIM_ElementAllocatedFromPool for allocated virtual processors

936 Table 5 lists the requirements for elements of this association. These requirements are in addition to
 937 those specified in the CIM Schema and in [DSP1041](#).

938 **Table 5 – Association: CIM_ElementAllocatedFromPool**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Value shall reference the instance of the CIM_ResourcePool class that represents a processor resource pool. Cardinality: 1
Dependent	Mandatory	Key: Value shall reference the instance of the CIM_Processor class that represents virtual processor resulting from a processor allocation from the pool. Cardinality: *

939 10.3 CIM_ElementAllocatedFromPool for resource pool hierarchies

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

942 Condition: Resource pool management (see 7.7) is implemented.

943 Table 6 lists the requirements for elements of this association. These requirements are in addition to
 944 those specified in the CIM Schema and in [DSP1041](#).

945

Table 6 – Association: CIM_ElementSettingData

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

946 **10.4 CIM_ElementSettingData for processor resource allocation**

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

949

Table 7 – Association: CIM_ElementSettingData for processor resource allocation

Elements	Requirement	Notes
ManagedElement	Mandatory	Key: Value shall reference the RASD instance represents the processor resource allocation. Cardinality: 1
SettingData	Mandatory	Key: Value shall reference the RASD instance that represents the processor resource allocation request. Cardinality: 0..1
IsDefault	Mandatory	Value shall be 1 (Is Default).

950 **10.5 CIM_ElementSettingData for processor resource pool**

951 The implementation of the CIM_ElementSettingData class for the representation of the relationship
952 between a child resource pool and its processor resource allocation is conditional.

953 Condition: Processor resource pool hierarchies (see 7.5) are implemented.

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

956

Table 8 – Association: CIM_ElementSettingData (Processor Resource Pool)

Elements	Requirement	Notes
ManagedElement	Mandatory	Key: Value shall reference the CIM_ResourcePool instance that represents the concrete processor resource pool. Cardinality: 0..1
SettingData	Mandatory	Key: Value shall reference the RASD instance that represents the processor resource allocation. Cardinality: 0..1

957 10.6 CIM_HostedDependency

958 The support of the CIM_HostedDependency association is optional.

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

961 **Table 9 – Association: CIM_HostedDependency**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: Value shall reference the CIM_Processor instance that represents host processor. Cardinality: 0..1
Dependent	Mandatory	Key: Value shall reference the CIM_Processor instance that represents virtual processor. Cardinality: 0..1

962 10.7 CIM_Processor (host processor)

963 The implementation of the CIM_Processor class for the representation of host processors is conditional.

964 Condition: The representation of host resources is implemented; see 7.2.

965 Table 10 lists the requirements for elements of this class. These requirements are in addition to those
966 specified in the CIM Schema and in [DSP1022](#) if that is implemented.

967 **Table 10 – Class: CIM_Processor (host processor)**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	Key
CreationClassName	Mandatory	Key
SystemName	Mandatory	Key
Name	Mandatory	Key
EnabledState	Mandatory	See CIM schema description.
RequestedState	Mandatory	See CIM schema description.

968 10.8 CIM_Processor (virtual processor)

969 See 7.9 for detailed implementation requirements for this class adaptation.

970 Table 11 lists the requirements for elements of this class adaptation. These requirements are in addition
971 to those specified in the CIM Schema and in [DSP1022](#) if that is implemented.

972

Table 11 – Class: CIM_Processor (virtual system)

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	Key
CreationClassName	Mandatory	Key
SystemName	Mandatory	Key
Name	Mandatory	Key
EnabledState	Mandatory	Unspecified.
RequestedState	Mandatory	Unspecified.

973 **10.9 CIM_RegisteredProfile**

974 The basic adaptation of the CIM_RegisteredProfile class is specified by [DSP1033](#).

975 Table 12 lists the requirements for elements of this class. These requirements are in addition to those
 976 specified in [DSP1033](#).

977

Table 12 – Class: CIM_RegisteredProfile

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

978 **10.10 CIM_ResourceAllocationFromPool**

979 The support of the CIM_ResourceAllocationFromPool association is optional.

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

982

Table 13 – Association: CIM_ResourceAllocationFromPool

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

983 **10.11 CIM_ResourceAllocationSettingData**

984 See 7.8.3 for detailed implementation requirements for this class.

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

987 **Table 14 – Class: CIM_ResourceAllocationSettingData**

Elements	Requirement	Notes
InstanceID	Mandatory	Key ; see DSP1041 .
ResourceType	Mandatory	Value shall be 3 (Processor).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 7.8.3.12 .
PoolID	Mandatory	See 7.8.3.2.
ConsumerVisibility	Optional	See 7.8.3.3.
HostResource[]	Optional	See 7.8.3.4.
AllocationUnits	Mandatory	See 7.8.3.5 .
VirtualQuantity	Mandatory	See 7.8.3.6.
Reservation	Optional	See 7.8.3.7.
Limit	Optional	See 7.8.3.8.
Weight	Optional	See 7.8.3.9.
AutomaticAllocation	Optional	See DSP1041 .
AutomaticDeallocation	Optional	See DSP1041 .
MappingBehavior	Optional	See 7.8.3.10.
VirtualQuantityUnits	Optional	See 7.8.3.11

988 10.12 CIM_ResourcePool

989 Table 15 lists the requirements for elements of this class. These requirements are in addition to those
 990 specified in the CIM Schema and in [DSP1041](#).

991 **Table 15 – Class: CIM_ResourcePool**

Elements	Requirement	Notes
InstanceID	Mandatory	Key
ElementName	Optional	See DSP1041 .
PoolID	Mandatory	See 7.3.3.
Primordial	Mandatory	See 7.3.4.
Capacity	Conditional	See 7.3.6.
Reserved	Optional	See 7.3.5.
ResourceType	Mandatory	Value shall be 3 (Processor).
OtherResourceType	Mandatory	Value shall be NULL.
ResourceSubType	Optional	See 7.3.2.
AllocationUnits	Mandatory	See 7.1

992 **10.13 CIM_SettingsDefineState**

993 Table 16 lists the requirements for elements of this association. These requirements are in addition to
 994 those specified in the CIM Schema and in [DSP1041](#).

995 **Table 16 – Association: CIM_SettingsDefineState**

Elements	Requirement	Notes
ManagedElement	Mandatory	Key: Value shall reference an instance of the CIM_Processor class. Cardinality: 0..1
SettingData	Mandatory	Key: Value shall reference the instance of the CIM_ResourceAllocationSettingData class. Cardinality: 0..1

996 **10.14 CIM_SystemDevice for host processor**

997 The implementation of the CIM_SystemDevice association for the representation of the relationship
 998 between host processors and their system is conditional.

999 Condition: The representation of host resources is implemented; see 7.2.

1000 NOTE If [DSP1022](#) is implemented for host processors, the implementation of the CIM_SystemDevice
 1001 association for the representation of the relationship between host processors and their system is required.

1002 If the CIM_SystemDevice association is implemented for the representation of the relationship between
 1003 host processors and their systems, the provisions in this subclause apply.

1004 Table 17 lists the requirements for elements of this association. These requirements are in addition to
 1005 those specified in the CIM Schema, in [DSP1041](#), and in [DSP1022](#) if that is implemented.

1006 **Table 17 – Association: CIM_SystemDevice (Host Processor)**

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

1007 **10.15 CIM_SystemDevice for virtual processor**

1008 Table 18 lists the requirements for elements of this association. These requirements are in addition to
1009 those specified in the CIM Schema and in [DSP1041](#).

1010 **Table 18 – Association: CIM_SystemDevice (Virtual Processor)**

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

1011

**ANNEX A
(Informative)****Change Log**1012
1013
1014
1015

Version	Date	Description
1.0.0a	2009-09-14	Release as work in progress
1.0.0c	2010-02-02	Release as work in progress
1.0.0	2010-04-22	Release as DMTF Standard

1016
1017