



1

2

3

4

**Document Number: DSP1022**

**Date: 2016-04-14**

**Version: 1.1.0**

5

## **CPU Profile**

6

**Supersedes: 1.0.2**

7

**Document Class: Normative**

8

**Document Status: Published**

9

**Document Language: en-US**

10 Copyright Notice

11 Copyright © 2008–2016 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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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>.

32 This document's normative language is English. Translation into other languages is permitted.

# CONTENTS

34	Foreword .....	6
35	Introduction.....	7
36	1 Scope .....	9
37	2 Normative references .....	9
38	3 Terms and definitions .....	9
39	4 Symbols and abbreviated terms.....	10
40	5 Synopsis .....	10
41	6 Description .....	11
42	7 Implementation.....	12
43	7.1 CIM_Processor .....	12
44	7.2 Processor capabilities .....	12
45	7.3 Processor state management.....	14
46	7.4 CIM_Processor.RequestedState .....	14
47	7.5 Modeling the current enabled state of the processor .....	15
48	7.6 Modeling individual processor cores.....	16
49	7.7 Modeling individual hardware threads .....	18
50	7.8 Modeling cache memory.....	20
51	7.9 Modeling physical aspects of processor and cache memory .....	22
52	8 Methods.....	22
53	8.1 CIM_Processor.RequestStateChange( ) .....	22
54	8.2 CIM_ProcessorCore.RequestStateChange( ) .....	23
55	8.3 CIM_HardwareThread.RequestStateChange( ) .....	24
56	8.4 CIM_Memory.RequestStateChange( ) .....	25
57	8.5 Profile conventions for operations .....	25
58	8.6 CIM_AssociatedCacheMemory .....	26
59	8.7 CIM_ConcreteComponent — References CIM_HardwareThread and CIM_Processor .....	26
60	8.8 CIM_ConcreteComponent — References CIM_ProcessorCore and CIM_Processor .....	26
61	8.9 CIM_ElementCapabilities — References CIM_HardwareThread and CIM_EnabledLogicalElementCapabilities.....	27
63	8.10 CIM_ElementCapabilities — References CIM_Memory and CIM_EnabledLogicalElementCapabilities.....	27
65	8.11 CIM_ElementCapabilities — References CIM_Processor and CIM_ProcessorCapabilities .....	27
68	8.12 CIM_ElementCapabilities — References CIM_ProcessorCore and CIM_EnabledLogicalElementCapabilities.....	28
69	8.13 CIM_EnabledLogicalElementCapabilities.....	28
70	8.14 CIM_HardwareThread .....	28
71	8.15 CIM_Memory .....	29
72	8.16 CIM_Processor .....	29
73	8.17 CIM_ProcessorCapabilities .....	29
74	8.18 CIM_ProcessorCore .....	30
75	8.19 CIM_SystemDevice .....	30
76	9 Use cases.....	31
77	9.1 Object diagrams.....	31
78	9.2 Change the enabled state of the memory to the desired state.....	36
79	9.3 Change the enabled state of the CPU to the desired state .....	37
80	9.4 Change the enabled state of the CPU's core to the desired state .....	37
81	9.5 Change the enabled state of the CPU's hardware thread to the desired state .....	37
82	9.6 Retrieve all the processor cores for the CPU .....	37
83	9.7 Retrieve all the hardware threads for the CPU .....	38
84	9.8 Retrieve CPU's cache memory information for the CPU.....	38

85	10	CIM Elements .....	38
86	10.1	CIM_AssociatedCacheMemory .....	39
87	10.2	CIM_ConcreteComponent — References CIM_HardwareThread and CIM_ProcessorCore .....	40
89	10.3	CIM_ConcreteComponent — References CIM_ProcessorCore and CIM_Processor .....	40
90	10.4	CIM_ElementCapabilities — References CIM_HardwareThread and CIM_EnabledLogicalElementCapabilities .....	40
92	10.5	CIM_ElementCapabilities — References CIM_Memory and CIM_EnabledLogicalElementCapabilities .....	41
94	10.6	CIM_ElementCapabilities — References CIM_Processor and CIM_ProcessorCapabilities .....	41
96	10.7	CIM_ElementCapabilities — References CIM_ProcessorCore and CIM_EnabledLogicalElementCapabilities .....	41
98	10.8	CIM_EnabledLogicalElementCapabilities .....	42
99	10.9	CIM_HardwareThread .....	42
100	10.10	CIM_Memory .....	42
101	10.11	CIM_Processor .....	43
102	10.12	CIM_ProcessorCapabilities .....	44
103	10.13	CIM_ProcessorCore .....	44
104	10.14	CIM_RegisteredProfile .....	45
105	10.15	CIM_SystemDevice .....	45
106		ANNEX A (informative) Change log .....	46
107			

## 108 Figures

109	Figure 1 – CPU Profile: Class Diagram .....	12
110	Figure 2 – Registered Profile .....	31
111	Figure 3 – Multi-Core CPU .....	32
112	Figure 4 – Detailed Multi-Core CPU .....	33
113	Figure 5 – Multi-Core CPU with a Disabled Core .....	34
114	Figure 6 – Single Core, Multi-Hardware Thread CPU .....	35
115	Figure 7 – Processor with Off-Chip Cache .....	36
116		

## 117 Tables

118	Table 1 – Related profiles .....	10
119	Table 2 – CIM_ProcessorCapabilities Properties Mapping to SMBIOS Equivalence .....	13
120	Table 3 – CIM_Processor.CPUStatus Value Descriptions .....	15
121	Table 4 – Mapping for CPUStatus Property and EnabledState Property Values .....	15
122	Table 5 – CIM_ProcessorCore.CoreEnabledState Value Descriptions .....	17
123	Table 6 – Mapping for the CoreEnabledState Property and EnabledState Property Values .....	18
124	Table 7 – CIM_HardwareThread.EnabledState Value Descriptions .....	20
125	Table 8 – CIM_Memory.EnabledState Value Descriptions .....	22
126	Table 9 – CIM_Processor.RequestStateChange( ) method: Return code values .....	23
127	Table 10 – CIM_Processor.RequestStateChange( ) method: Parameters .....	23
128	Table 11 – CIM_ProcessorCore.RequestStateChange( ) method: Return code values .....	23
129	Table 12 – CIM_ProcessorCore.RequestStateChange( ) method: Parameters .....	24
130	Table 13 – CIM_HardwareThread.RequestStateChange( ) method: Return code values .....	24

131 Table 14 – CIM\_HardwareThread.RequestStateChange( ) method: Parameters..... 24

132 Table 15 – CIM\_Memory.RequestStateChange( ) method: Return code values ..... 25

133 Table 16 – CIM\_Memory.RequestStateChange( ) method: Parameters..... 25

134 Table 17 – Operations: CIM\_AssociatedCacheMemory..... 26

135 Table 18 – Operations: CIM\_ConcreteComponent ..... 26

136 Table 19 – Operations: CIM\_ConcreteComponent ..... 27

137 Table 20 – Operations: CIM\_ElementCapabilities ..... 27

138 Table 21 – Operations: CIM\_ElementCapabilities ..... 27

139 Table 22 – Operations: CIM\_ElementCapabilities ..... 28

140 Table 23 – Operations: CIM\_ElementCapabilities ..... 28

141 Table 24 – Operations: CIM\_HardwareThread..... 28

142 Table 25 – Operations: CIM\_Memory..... 29

143 Table 26 – Operations: CIM\_Processor..... 29

144 Table 27 – Operations: CIM\_ProcessorCore..... 30

145 Table 28 – Operations: CIM\_SystemDevice..... 30

146 Table 29 – CIM Elements: CPU Profile..... 38

147 Table 30 – Class: CIM\_AssociatedCacheMemory ..... 39

148 Table 31 – Class: CIM\_ConcreteComponent — References CIM\_HardwareThread and  
149 CIM\_ProcessorCore..... 40

150 Table 32 – Class: CIM\_ConcreteComponent — References CIM\_ProcessorCore and CIM\_Processor .. 40

151 Table 33 – Class: CIM\_ElementCapabilities — References CIM\_HardwareThread and  
152 CIM\_EnabledLogicalElementCapabilities ..... 41

153 Table 34 – Class: CIM\_ElementCapabilities — References CIM\_Memory and  
154 CIM\_EnabledLogicalElementCapabilities ..... 41

155 Table 35 – Class: CIM\_ElementCapabilities — References CIM\_Processor and  
156 CIM\_ProcessorCapabilities..... 41

157 Table 36 – Class: CIM\_ElementCapabilities — References CIM\_ProcessorCore and  
158 CIM\_EnabledLogicalElementCapabilities ..... 42

159 Table 37 – Class: CIM\_EnabledLogicalElementCapabilities..... 42

160 Table 38 – Class: CIM\_HardwareThread ..... 42

161 Table 39 – Class: CIM\_Memory ..... 43

162 Table 40 – Class: CIM\_Processor ..... 43

163 Table 41 – Class: CIM\_ProcessorCapabilities..... 44

164 Table 42 – Class: CIM\_ProcessorCore ..... 44

165 Table 43 – Class: CIM\_RegisteredProfile..... 45

166 Table 44 – Class: CIM\_SystemDevice ..... 45

167

168

## Foreword

169 The *CPU Profile* (DSP1022) was prepared by the Server Desktop Mobile Platforms Working Group of the  
170 DMTF.

171 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems  
172 management and interoperability. For information about the DMTF, see <http://www.dmf.org>.

### 173 **Acknowledgments**

174 The DMTF acknowledges the following individuals for their contributions to this document:

175 Editors:

- 176 • Jon Hass – Dell
- 177 • Jeff Hilland – Hewlett-Packard Company
- 178 • John Leung – Intel
- 179 • Khachatur Papanyan – Dell

180 Contributors:

- 181 • Jeff Lynch – IBM
- 182 • Aaron Merkin – IBM
- 183 • Khachatur Papanyan – Dell
- 184 • Christina Shaw – Hewlett-Packard Company
- 185 • Perry Vincent – Intel

186

## Introduction

187 The information in this specification should be sufficient for a provider or consumer of this data to identify  
188 unambiguously the classes, properties, methods, and values that shall be instantiated and manipulated to  
189 represent and manage the processor components of systems and subsystems modeled using the DMTF  
190 Common Information Model (CIM) core and extended model definitions.

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

### 193 Document conventions

#### 194 Experimental material

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

200 The following typographical convention indicates experimental material:

---

---

#### 201 **EXPERIMENTAL**

202 Experimental material appears here.

#### 203 **EXPERIMENTAL**

---

---

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

206

207

208



209

# CPU Profile

## 210 1 Scope

211 The *CPU Profile* extends the management capability of referencing profiles by adding the capability to  
212 represent CPUs or processors in a managed system. CPU cache memory and associations with CPU  
213 physical aspects, as well as profile implementation version information, are modeled in this profile.

## 214 2 Normative references

215 The following referenced documents are indispensable for the application of this document. For dated or  
216 versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies.  
217 For references without a date or version, the latest published edition of the referenced document  
218 (including any corrigenda or DMTF update versions) applies.

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

221 DMTF DSP0134, *System Management BIOS (SMBIOS) Reference Specification 2.6*,  
222 [http://www.dmtf.org/standards/published\\_documents/DSP0134\\_2.6.pdf](http://www.dmtf.org/standards/published_documents/DSP0134_2.6.pdf)

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

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

227 DMTF DSP1011, *Physical Asset Profile 1.0*,  
228 [http://www.dmtf.org/standards/published\\_documents/DSP1011\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1011_1.0.pdf)

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

231 IETF RFC5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008,  
232 <http://www.rfc-editor.org/rfc/rfc5234.txt>

233 ISO/IEC Directives, Part 2, [Rules for the structure and drafting of International Standards](#)

## 234 3 Terms and definitions

235 In this document, some terms have a specific meaning beyond the normal English meaning. Those terms  
236 are defined in this clause.

237 The terms "shall" ("required"), "shall not," "should" ("recommended"), "should not" ("not recommended"),  
238 "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described  
239 in [ISO/IEC Directives, Part 2](#), Annex H. The terms in parenthesis are alternatives for the preceding term,  
240 for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that  
241 [ISO/IEC Directives, Part 2](#), Annex H specifies additional alternatives. Occurrences of such additional  
242 alternatives shall be interpreted in their normal English meaning.

243 The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as  
244 described in [ISO/IEC Directives, Part 2](#), Clause 5.

245 The terms "normative" and "informative" in this document are to be interpreted as described in [ISO/IEC](#)  
 246 [Directives, Part 2](#), Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do  
 247 not contain normative content. Notes and examples are always informative elements.

248 For the purposes of this document, the following terms and definitions apply. The terms defined in  
 249 [DSP0004](#), [DSP0200](#), [DSP1001](#), and [DSP1033](#) also apply to this document.

### 250 3.1

#### 251 Cache Memory

252 indicates the instance of CIM\_Memory that represents the cache memory for the processor

### 253 3.2

#### 254 Host Processor

255 indicates the instance of CIM\_Processor that represents the processor that hosts the processor core

### 256 3.3

#### 257 Threading Processor Core

258 indicates the instance of CIM\_ProcessorCore that represents the processor core that enables the  
 259 hardware threading

## 260 4 Symbols and abbreviated terms

### 261 4.1

#### 262 CPU

263 central processing unit

## 264 5 Synopsis

265 **Profile name:** CPU Profile

266 **Version:** 1.1.0

267 **Organization:** DMTF

268 **CIM Schema version:** 2.45

269 **Central class:** CIM\_Processor

270 **Scoping class:** CIM\_ComputerSystem

271 The *CPU Profile* is a component profile that extends the management capability of referencing profiles by  
 272 adding the capability to represent CPUs or processors in a managed system.

273

**Table 1 – Related profiles**

Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0	Optional	See 7.9.
Profile Registration	DMTF	1.0	Mandatory	

## 274 6 Description

275 The *CPU Profile* describes CPU or processor devices and associated cache memory used in managed  
276 systems.

277 The profile could manage the following capabilities of a typical computer system:

- 278 • A computer system can have one or more processors, which may be individually enabled or  
279 disabled.
- 280 • A processor can contain one or more processor cores, which may be individually enabled or  
281 disabled.
- 282 • A processor core can contain one or more hardware threads, which may be individually enabled or  
283 disabled

284 Figure 1 represents the class schema for the *CPU Profile*. For simplicity, the prefix CIM\_ has been  
285 removed from the names of the classes.

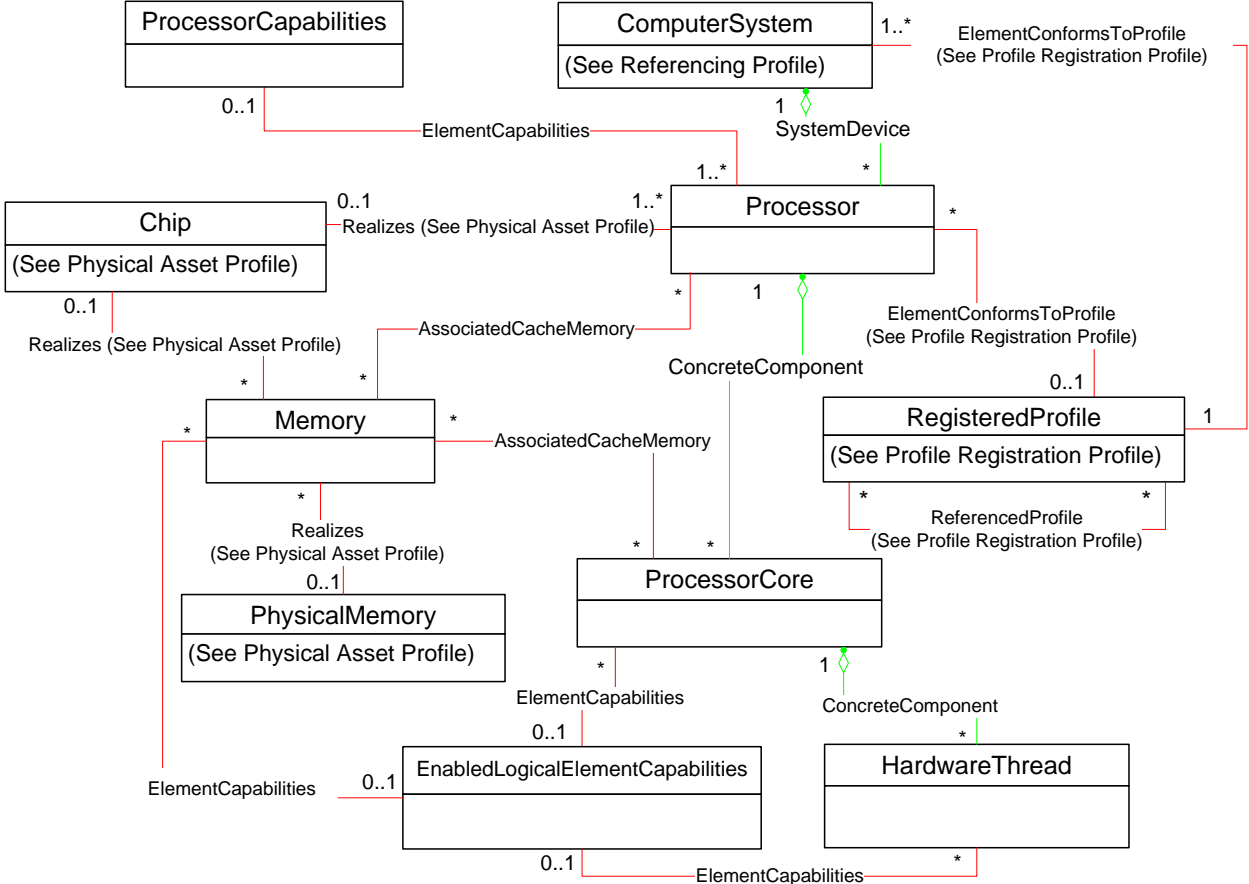
286 The CIM\_Processor class represents a group of processor cores; the CIM\_ProcessorCapabilities class  
287 describes the capabilities of the processor. The CIM\_Processor may be associated to one or more of  
288 instances of CIM\_ProcessorCore, through the CIM\_ConcreteComponent association.

289 The CIM\_ProcessorCore class represents a processing execution unit. The CIM\_ProcessorCore may be  
290 associated to one or more instances of CIM\_HardwareThread, through the CIM\_ConcreteComponent  
291 association.

292 The CIM\_HardwareThread class represents a hardware thread, a mechanism by which a processing  
293 execute unit is made to appear as multiple processing units (each called a virtual core).

294 The CIM\_Memory class represents cache memory. CIM\_Memory may be associated to either  
295 CIM\_Processor or CIM\_ProcessorCore, through the CIM\_AssociatedCacheMemory association.

296 The CIM\_Chip class represents the physical aspects of a processor. The CIM\_PhysicalMemory  
297 represents the cache memory, when the cache memory is off-chip/external.



298

299

Figure 1 – CPU Profile: Class Diagram

300 7 Implementation

301 This clause details the requirements related to the arrangement of instances and their properties for  
302 implementations of this profile. Methods are listed in clause 8 (“Methods”), and properties are listed in  
303 clause 10 (“CIM Elements”).

304 7.1 CIM\_Processor

305 Zero or more instances of CIM\_Processor shall be instantiated.

306 7.2 Processor capabilities

307 The CIM\_ProcessorCapabilities class may be instantiated to represent the processor capabilities. Only  
308 one instance of CIM\_ProcessorCapabilities shall be associated with a given instance of CIM\_Processor  
309 through an instance of CIM\_ElementCapabilities.

310 **7.2.1 Multi-Core or Multi-Thread processor capabilities**

311 When modeling the capabilities of a multi-core or multi-thread processor, the CIM\_ProcessorCapabilities  
 312 class shall be instantiated and associated to the instance of CIM\_Processor that represents the multi-core  
 313 or multi-thread processor.

314 The properties of CIM\_ProcessorCapabilities described in the “CIM\_ProcessorCapabilities Properties”  
 315 column in Table 2 are defined in terms of the [DSP0134](#) Processor Information structure to provide  
 316 meaningful context for the interpretation of the properties. The mappings specified in Table 2 shall be  
 317 used. The underlying represented system does not need to support [DSP0134](#).

318 **Table 2 – CIM\_ProcessorCapabilities Properties Mapping to SMBIOS Equivalence**

CIM_ProcessorCapabilities Properties	SMBIOS Structure Name	SMBIOS Structure Description
NumberOfProcessorCores	Core Count	Number of cores per processor socket
NumberOfHardwareThreads	Thread Count	Number of threads per processor socket

319 **7.2.2 Single-Core and Single-Thread processor capabilities**

320 When modeling the capabilities of a single-core and single-thread processor, the  
 321 CIM\_ProcessorCapabilities may not be instantiated.

322 When no instance of CIM\_ProcessorCapabilities is associated with the instance of CIM\_Processor that  
 323 represents the processor, the processor is a single-core and single-thread processor.

324 When an instance of CIM\_ProcessorCapabilities is associated with the instance of CIM\_Processor that  
 325 represents the single-core and single-thread processor, the following requirements apply:

- 326 • The CIM\_ProcessorCapabilities.NumberOfProcessorCores property shall have a value of 1.
- 327 • The CIM\_ProcessorCapabilities.NumberOfHardwareThreads property shall have a value of 1.

328 **7.2.3 CIM\_ProcessorCapabilities.RequestedStatesSupported**

329 The RequestedStatesSupported property is an array that contains the supported requested states for the  
 330 instance of CIM\_Processor. This property shall be the super set of the values to be used as the  
 331 RequestedState parameter in the RequestStateChange() method (see 8.1). The value of the  
 332 CIM\_ProcessorCapabilities.RequestedStatesSupported property shall be an empty array or contain any  
 333 combination of the following values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

334 **7.2.4 CIM\_ProcessorCapabilities.ElementNameEditSupported**

335 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports  
 336 client modification of the CIM\_Processor.ElementName property.

337 **7.2.5 CIM\_ProcessorCapabilities.MaxElementNameLen**

338 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
 339 property has a value of TRUE.

---

340 **EXPERIMENTAL**

341 **7.2.6 CIM\_ProcessorCapabilities.ProcessorArchitecture**

342 The ProcessorArchitecture property may contain a value that represents the processor architecture.

### 343 **7.2.7 CIM\_ProcessorCapabilities.InstructionSet**

344 The InstructionSet property may contain a value that represents the instruction set supported by the  
345 processor.

### 346 **7.2.8 CIM\_ProcessorCapabilities.InstructionSetExtensionName**

347 The InstructionSetExtensionName indexed array property may contain values that represents the  
348 instruction set extensions supported by the processor.

### 349 **7.2.9 CIM\_ProcessorCapabilities.InstructionSetExtensionStatus**

350 The InstructionSetExtensionStatus indexed array property may contain values that represents the state of  
351 the corresponding instruction set extension. The values shall be either "Enabled", "Disabled" or  
352 "Unknown".

## 353 **EXPERIMENTAL**

---

## 354 **7.3 Processor state management**

355 Processor state management requires that the CIM\_Processor.RequestStateChange( ) method be  
356 supported (see 8.1) and the value of the CIM\_Processor.RequestedState property not match 12 (Not  
357 Applicable).

### 358 **7.3.1 Processor state management support**

359 When the instance of CIM\_ProcessorCapabilities does not exist, processor state management shall not  
360 be supported.

361 When the value of the CIM\_ProcessorCapabilities.RequestedStatesSupported property of the associated  
362 CIM\_ProcessorCapabilities instance is an empty array, processor state management shall not be  
363 supported.

364 When the value of the CIM\_ProcessorCapabilities.RequestedStatesSupported property of the associated  
365 CIM\_ProcessorCapabilities instance is not an empty array, processor state management shall be  
366 supported.

## 367 **7.4 CIM\_Processor.RequestedState**

368 The CIM\_Processor.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No Change),  
369 or a value contained in the CIM\_ProcessorCapabilities.RequestedStatesSupported property array of the  
370 associated CIM\_ProcessorCapabilities instance (see 7.2.2).

371 When processor state management is supported and the RequestStateChange( ) method is successfully  
372 executed, the RequestedState property shall be set to the value of the RequestedState parameter of the  
373 RequestStateChange( ) method. After the RequestStateChange( ) method has successfully executed, the  
374 RequestedState and EnabledState properties shall have equal values with the exception of the  
375 transitional requested state 11 (Reset). The value of the RequestedState property may also change as a  
376 result of a request for a change to the processor's enabled state by a non-CIM implementation.

### 377 **7.4.1 RequestedState — 12 (Not Applicable) value**

378 When processor state management is not supported, the value of the CIM\_Processor.RequestedState  
379 property shall be 12 (Not Applicable).

380 **7.4.2 RequestedState — 5 (No Change) value**

381 When processor state management is supported, the initial value of the CIM\_Processor.RequestedState  
 382 property shall be 5 (No Change).

383 **7.5 Modeling the current enabled state of the processor**

384 The current enabled state of the processor is described by the CIM\_Processor.CPUStatus and  
 385 CIM\_Processor.EnabledState properties. Clauses 7.5.1 and 7.5.2 detail the requirements for  
 386 implementing these two properties.

387 **7.5.1 CIM\_Processor.CPUStatus**

388 Table 3 describes the mapping between the values of the CIM\_Processor.CPUStatus property and the  
 389 corresponding description of the state of the processor. The CPUStatus property shall match the values  
 390 that are specified in Table 3. When the RequestStateChange() method executes but does not complete  
 391 successfully, or the processor is in an indeterminate state, the CPUStatus property shall have value of 0  
 392 (Unknown). The value of this property may also change as a result of a change to the processor's  
 393 enabled state by a non-CIM implementation.

394 **Table 3 – CIM\_Processor.CPUStatus Value Descriptions**

Value	Description	Extended Description
0	Unknown	Processor state is indeterminate, or the processor state management is not supported.
1	CPU Enabled	Processor shall be enabled.
2	CPU Disabled by User	Processor shall be disabled through client configuration, which may occur through client invocation of the RequestStateChange() method or through a non-CIM implementation such as BIOS.
3	CPU Disabled By BIOS (POST Error)	Processor shall be disabled due to a POST error.
4	CPU Is Idle, waiting to be enabled	Processor shall be enabled but idling.

395 **7.5.2 CIM\_Processor.EnabledState**

396 The CIM\_Processor.EnabledState property shall be implemented in addition to the  
 397 CIM\_Processor.CPUStatus property. When the CPUStatus property has a value that matches a value in  
 398 the "CPUStatus Value" column in Table 4, the EnabledState property shall have a value that matches a  
 399 value in the "EnabledState Value" column in the same row in the table.

400 **Table 4 – Mapping for CPUStatus Property and EnabledState Property Values**

CPUSatus Value	Description	EnabledState Value	Description
0	Unknown	0 or 5	Unknown or Not Applicable
1	CPU Enabled	2	Enabled
2	CPU Disabled by User	3	Disabled
3	CPU Disabled By BIOS (POST Error)	3	Disabled
4	CPU Is Idle, waiting to be enabled	2	Enabled

## 401 **7.6 Modeling individual processor cores**

402 Modeling the individual processor cores is optional functionality. When individual processor cores are  
403 modeled, the implementation shall instantiate an instance of CIM\_ProcessorCore to represent each  
404 processor core. All the requirements in this clause and its subclasses are applicable when an  
405 implementation instantiates the CIM\_ProcessorCore class.

406 Each instance of CIM\_ProcessorCore shall be associated through an instance of  
407 CIM\_ConcreteComponent to only one instance of CIM\_Processor that represents the processor (Host  
408 Processor) that hosts the processor core.

409 The number of instances of CIM\_ProcessorCore associated with the Host Processor shall be equal to the  
410 value of the CIM\_ProcessorCapabilities.NumberOfProcessorCores property of the instance of  
411 CIM\_ProcessorCapabilities that is associated with the Host Processor.

### 412 **7.6.1 Processor core capabilities**

413 The CIM\_EnabledLogicalElementCapabilities class may be used to model the capabilities of processor  
414 cores. When the CIM\_EnabledLogicalElementCapabilities class is instantiated to represent the processor  
415 core capabilities, the instance of CIM\_EnabledLogicalElementCapabilities shall be associated with the  
416 CIM\_ProcessorCore instance through an instance of CIM\_ElementCapabilities and used for advertising  
417 the capabilities of the CIM\_ProcessorCore instance.

418 There shall be at most one instance of CIM\_EnabledLogicalElementCapabilities associated with a given  
419 instance of CIM\_ProcessorCore.

#### 420 **7.6.1.1 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported**

421 The RequestedStatesSupported property is an array that contains the supported requested states for the  
422 instance of CIM\_ProcessorCore. This property shall be the super set of the values to be used as the  
423 RequestedState parameter in the RequestStateChange() method (see 8.2). The value of the  
424 RequestedStatesSupported property shall be an empty array or contain any combination of the following  
425 values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

#### 426 **7.6.1.2 CIM\_EnabledLogicalElementCapabilities.ElementNameEditSupported**

427 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports  
428 client modification of the CIM\_ProcessorCore.ElementName property.

#### 429 **7.6.1.3 CIM\_EnabledLogicalElementCapabilities.MaxElementNameLen**

430 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
431 property has a value of TRUE.

### 432 **7.6.2 Processor core state management**

433 Processor core state management requires that the CIM\_ProcessorCore.RequestStateChange() method  
434 be supported (see 8.2) and the value of the CIM\_ProcessorCore.RequestedState property not match 12  
435 (Not Applicable).

#### 436 **7.6.2.1 Processor core state management support**

437 When no CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_ProcessorCore  
438 instance, processor core state management shall not be supported.



439 When a CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_ProcessorCore  
 440 instance but the value of the CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported  
 441 property is an empty array, processor core state management shall not be supported.

442 When a CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_ProcessorCore  
 443 instance and the value of the CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported  
 444 property is not an empty array, processor core state management shall be supported.

445 **7.6.3 CIM\_ProcessorCore.RequestedState**

446 The CIM\_ProcessorCore.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No  
 447 Change), or a value contained in the  
 448 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated  
 449 CIM\_EnabledLogicalElementCapabilities instance (see 7.6.1.1).

450 When processor core state management is supported and the RequestStateChange() method is  
 451 successfully executed, the RequestedState property shall be set to the value of the RequestedState  
 452 parameter of the RequestStateChange() method. After the RequestStateChange() method has  
 453 successfully executed, the RequestedState and EnabledState properties shall have equal values with the  
 454 exception of the transitional requested state 11 (Reset). The value of the RequestedState property may  
 455 also change as a result of a request for a change to the processor’s enabled state by a non-CIM  
 456 implementation.

457 **7.6.3.1 RequestedState — 12 (Not Applicable) value**

458 When processor core state management is not supported, the value of the  
 459 CIM\_ProcessorCore.RequestedState property shall be 12 (Not Applicable).

460 **7.6.3.2 RequestedState — 5 (No Change) value**

461 When processor core state management is supported, the initial value of the  
 462 CIM\_ProcessorCore.RequestedState property shall be 5 (No Change).

463 **7.6.4 Modeling the current enabled state of the processor core**

464 The current enabled state of the processor core is described by the  
 465 CIM\_ProcessorCore.CoreEnabledState and CIM\_ProcessorCore.EnabledState properties. Clauses  
 466 7.6.4.1 and 7.6.4.2 detail the requirements for implementing these two properties.

467 **7.6.4.1 CIM\_ProcessorCore.CoreEnabledState**

468 Table 5 describes the mapping between the values of the CIM\_ProcessorCore.CoreEnabledState  
 469 property and the corresponding description of the state of the processor core. The CoreEnabledState  
 470 property shall match the values that are specified in Table 5. When the RequestStateChange() method  
 471 executes but does not complete successfully, and the processor core is in an indeterminate state, the  
 472 CoreEnabledState property shall have a value of 0 (Unknown). The value of this property may also  
 473 change as a result of a change to the processor’s enabled state by a non-CIM implementation.

474 **Table 5 – CIM\_ProcessorCore.CoreEnabledState Value Descriptions**

Value	Description	Extended Description
0	Unknown	Processor core state is indeterminate, or the processor core state management is not supported.
2	Enabled	Processor core shall be enabled.
3	Disabled	Processor core shall be disabled.

Value	Description	Extended Description
4	Core Disabled User	Processor core shall be disabled through client configuration, which may occur through client invocation of RequestStateChange() or through a non-CIM implementation such as BIOS.
5	Core Disabled By Post Error	Processor core shall be disabled due to a POST error.

#### 475 7.6.4.2 CIM\_ProcessorCore.EnabledState

476 The CIM\_ProcessorCore.EnabledState property shall be implemented in addition to the  
 477 CIM\_ProcessorCore.CoreEnabledState property. When the CoreEnabledState property value matches a  
 478 value in the “CoreEnabledState Value” column in Table 6, the EnabledState property shall have the value  
 479 that matches the value in the “EnabledState Value” column in the same row in the table.

480 **Table 6 – Mapping for the CoreEnabledState Property and EnabledState Property Values**

CoreEnabledState Value	Description	EnabledState Value	Description
0	Unknown	0 or 5	Unknown or Not Applicable
2	Enabled	2	Enabled
3	Disabled	3	Disabled
4	Core Disabled User	3	Disabled
5	Core Disabled By Post Error	3	Disabled

### 481 7.7 Modeling individual hardware threads

482 Modeling the individual hardware threads is optional functionality. When hardware threads are modeled,  
 483 the implementation shall model processor cores as described in 7.6 and shall instantiate an instance of  
 484 CIM\_HardwareThread to represent each hardware thread. All the requirements in this clause and its  
 485 subclauses are applicable when an implementation instantiates the CIM\_HardwareThread class.

486 The instance of CIM\_HardwareThread shall be associated through an instance of  
 487 CIM\_ConcreteComponent to only one instance of CIM\_ProcessorCore that represents the processor core  
 488 that enables the hardware thread (Threading Processor Core).

489 For a given Host Processor, the number of instances of CIM\_HardwareThread that are associated with  
 490 Threading Processor Cores, which in turn are associated with the Host Processor, shall be equal to the  
 491 value of the NumberOfHardwareThreads property of the instance of CIM\_ProcessorCapabilities that is  
 492 associated with the Host Processor.

#### 493 7.7.1 Hardware thread capabilities

494 When the CIM\_EnabledLogicalElementCapabilities class is instantiated to represent the hardware thread  
 495 capabilities, the instance of CIM\_EnabledLogicalElementCapabilities shall be associated with the  
 496 CIM\_HardwareThread instance through an instance of CIM\_ElementCapabilities and used for advertising  
 497 the capabilities of the CIM\_HardwareThread instance.

498 At most one instance of CIM\_EnabledLogicalElementCapabilities shall be associated with a given  
 499 instance of CIM\_HardwareThread.

### 500 7.7.1.1 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported

501 The RequestedStatesSupported property is an array that contains the supported requested states for the  
502 instance of CIM\_HardwareThread. This property shall be the super set of the values to be used as the  
503 RequestedState parameter in the RequestStateChange() method (see 8.3). The value of the  
504 RequestedStatesSupported property shall be an empty array or contain any combination of the following  
505 values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

### 506 7.7.1.2 CIM\_EnabledLogicalElementCapabilities.ElementNameEditSupported

507 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports  
508 client modification of the CIM\_HardwareThread.ElementName property.

### 509 7.7.1.3 CIM\_EnabledLogicalElementCapabilities.MaxElementNameLen

510 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
511 property has a value of TRUE.

## 512 7.7.2 Hardware thread state management

513 Hardware thread state management requires that the CIM\_HardwareThread.RequestStateChange()  
514 method be supported (see 8.3) and the value of the CIM\_HardwareThread.RequestedState property not  
515 match 12 (Not Applicable).

### 516 7.7.2.1 Hardware thread state management support

517 When no CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_HardwareThread  
518 instance, hardware thread state management shall not be supported.

519 When a CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_HardwareThread  
520 instance but the value of the CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported  
521 property is an empty array, hardware thread state management shall not be supported.

522 When a CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_HardwareThread  
523 instance and the value of the CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported  
524 property is not an empty array, hardware thread state management shall be supported.

### 525 7.7.3 CIM\_HardwareThread.RequestedState

526 The CIM\_HardwareThread.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No  
527 Change), or a value contained in the  
528 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated  
529 CIM\_EnabledLogicalElementCapabilities instance (see 7.7.1.1).

530 When hardware thread state management is supported and the RequestStateChange() method is  
531 successfully executed, the RequestedState property shall be set to the value of the RequestedState  
532 parameter of the RequestStateChange() method. After the RequestStateChange() method has  
533 successfully executed, the RequestedState and EnabledState properties shall have equal values with the  
534 exception of the transitional requested state 11 (Reset). The value of the RequestedState property may  
535 also change as a result of a request for a change to the hardware thread's enabled state by a non-CIM  
536 implementation.

### 537 7.7.3.1 RequestedState — 12 (Not Applicable) value

538 When hardware thread state management is not supported, the value of the  
539 CIM\_HardwareThread.RequestedState property shall be 12 (Not Applicable).

### 540 7.7.3.2 RequestedState — 5 (No Change) value

541 When hardware thread state management is supported, the initial value of the  
542 CIM\_HardwareThread.RequestedState property shall be 5 (No Change).

### 543 7.7.4 CIM\_HardwareThread.EnabledState

544 Table 7 describes the mapping between the values of the CIM\_HardwareThread.EnabledState property  
545 and the corresponding description of the state of the hardware thread. The EnabledState property shall  
546 match the values that are specified in Table 7. When the RequestStateChange() method executes but  
547 does not complete successfully, and the hardware thread is in an indeterminate state, the EnabledState  
548 property shall have a value of 5 (Not Applicable). The value of this property may also change as a result  
549 of a change to the hardware thread's enabled state by a non-CIM implementation.

550 **Table 7 – CIM\_HardwareThread.EnabledState Value Descriptions**

Value	Description	Extended Description
2	Enabled	Hardware thread shall be enabled.
3	Disabled	Hardware thread shall be disabled.
5	Not Applicable	Hardware thread state is indeterminate, or hardware thread state management is not supported.

## 551 7.8 Modeling cache memory

552 Modeling the cache memory of the processor is optional. The implementation may instantiate instances of  
553 CIM\_Memory to represent the cache memory. All the requirements in this clause and its subclauses are  
554 applicable when an implementation instantiates the CIM\_Memory class that represents cache memory.

555 A single instance of CIM\_Memory shall exist for each discrete cache memory. When the cache memory is  
556 shared, the single instance of CIM\_Memory shall be associated with multiple instances of CIM\_Processor  
557 or CIM\_ProcessorCore. When the cache memory is not shared, the instance of CIM\_Memory shall be  
558 associated with exactly one instance of CIM\_Processor or CIM\_ProcessorCore.

559 When the optional behavior described in 7.6 is implemented, each instance of CIM\_Memory that  
560 represents the cache memory used by the processor core shall be associated with the instance of  
561 CIM\_ProcessorCore that represents the processor core through an instance of  
562 CIM\_AssociatedCacheMemory and shall not be associated with the Host Processor of the core.

563 When the optional behavior described in 7.6 is not implemented, each instance of CIM\_Memory that  
564 represents the cache memory used by the processor shall be associated through an instance of the  
565 CIM\_AssociatedCacheMemory to the instance of CIM\_Processor.

### 566 7.8.1 Cache memory capabilities

567 When the CIM\_EnabledLogicalElementCapabilities class is instantiated to represent the cache memory  
568 capabilities, the instance of CIM\_EnabledLogicalElementCapabilities shall be associated with the  
569 CIM\_Memory instance through an instance of CIM\_ElementCapabilities and used for advertising the  
570 capabilities of the CIM\_Memory instance.

571 At most one instance of CIM\_EnabledLogicalElementCapabilities shall be associated with a given  
572 instance of CIM\_Memory.

**573 7.8.1.1 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported**

574 The RequestedStatesSupported property is an array that contains the supported requested states for the  
575 instance of CIM\_Memory. This property shall be the super set of the values to be used as the  
576 RequestedState parameter in the RequestStateChange() method (see 8.4). The value of the  
577 RequestedStatesSupported property shall be an empty array or contain any combination of the following  
578 values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

**579 7.8.1.2 CIM\_EnabledLogicalElementCapabilities.ElementNameEditSupported**

580 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports  
581 client modification of the CIM\_Memory.ElementName property.

**582 7.8.1.3 CIM\_EnabledLogicalElementCapabilities.MaxElementNameLen**

583 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
584 property has a value of TRUE.

**585 7.8.2 Cache memory state management**

586 Cache memory state management requires that the CIM\_Memory.RequestStateChange() method be  
587 supported (see 8.4) and the value of the CIM\_Memory.RequestedState property not match 12 (Not  
588 Applicable).

**589 7.8.2.1 Cache memory state management support**

590 When no CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_Memory instance,  
591 cache memory state management shall not be supported.

592 When a CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_Memory instance  
593 but the value of the CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported property is an  
594 empty array, cache memory state management shall not be supported.

595 When a CIM\_EnabledLogicalElementCapabilities instance is associated with the CIM\_Memory instance  
596 and the value of the CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported property is not  
597 an empty array, cache memory state management shall be supported.

**598 7.8.3 CIM\_Memory.RequestedState**

599 The CIM\_Memory.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No Change),  
600 or a value contained in the CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported property  
601 array of the associated CIM\_EnabledLogicalElementCapabilities instance (see 7.8.1.1).

602 When cache memory state management is supported and the RequestStateChange() method is  
603 successfully executed, the RequestedState property shall be set to the value of the RequestedState  
604 parameter of the RequestStateChange() method. After the RequestStateChange() method has  
605 successfully executed, the RequestedState and EnabledState properties shall have equal values with the  
606 exception of the transitional requested state 11 (Reset). The value of the RequestedState property may  
607 also change as a result of a request for a change to the cache memory's enabled state by a non-CIM  
608 implementation.

**609 7.8.3.1 RequestedState — 12 (Not Applicable) value**

610 When cache memory state management is not supported, the value of the CIM\_Memory.RequestedState  
611 property shall be 12 (Not Applicable).

### 612 7.8.3.2 RequestedState — 5 (No Change) value

613 When cache memory state management is supported, the initial value of the  
614 CIM\_Memory.RequestedState property shall be 5 (No Change).

### 615 7.8.4 CIM\_Memory.EnabledState

616 Table 8 describes the mapping between the values of the CIM\_Memory.EnabledState property and the  
617 corresponding description of the state of the cache memory. The EnabledState property shall match the  
618 values that are specified in Table 8. When the RequestStateChange() method executes but does not  
619 complete successfully, and the cache memory is in an indeterminate state, the EnabledState property  
620 shall have value of 5 (Not Applicable). The value of this property may also change as a result of a change  
621 to the cache memory's enabled state by a non-CIM implementation.

622 **Table 8 – CIM\_Memory.EnabledState Value Descriptions**

Value	Description	Extended Description
2	Enabled	Cache memory shall be enabled.
3	Disabled	Cache memory shall be disabled.
5	Not Applicable	Cache memory state is indeterminate, or cache memory state management is not supported.

## 623 7.9 Modeling physical aspects of processor and cache memory

624 The [Physical Asset Profile](#) may be implemented to model the physical aspects of a processor, including  
625 the asset information of the processor or the internal or off-chip cache memory.

626 When the processor's or internal cache memory's physical aspects are represented, a CIM\_Chip instance  
627 shall be instantiated and associated with the instance of CIM\_Processor or with any instances of  
628 CIM\_Memory that represent the internal cache through instances of CIM\_Realizes.

629 When the off-chip cache memory is represented along with its physical aspects, a CIM\_PhysicalMemory  
630 instance shall be instantiated and associated with the instance of CIM\_Memory through an instance of  
631 CIM\_Realizes.

632 When processor cores or hardware threads for the processor are modeled with the physical aspects of  
633 the processor, the instances of CIM\_ProcessorCore and CIM\_HardwareThread shall not be associated  
634 with the instance of CIM\_Chip that represents the physical aspects of the processor.

635 The configuration capacity of the managed system for processors may be modeled using the optional  
636 behavior specified in the "Modeling Configuration Capacity" clause of the [Physical Asset Profile](#).

## 637 8 Methods

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

### 640 8.1 CIM\_Processor.RequestStateChange()

641 Invocation of the CIM\_Processor.RequestStateChange() method changes the element's state to the  
642 value that is specified in the RequestedState parameter.

643 Return code values for the RequestStateChange() method shall be as specified in Table 9. Parameters  
644 of the RequestStateChange() method are specified in Table 10.

- 645 When processor state management is supported, the RequestStateChange() method shall be
- 646 implemented and shall not return a value of 1 (Not Supported) (see 7.3.1).
- 647 Invoking the CIM\_Processor.RequestStateChange() method multiple times could result in earlier
- 648 requests being overwritten or lost.
- 649 No standard messages are defined for this method.

650 **Table 9 – CIM\_Processor.RequestStateChange() method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

651 **Table 10 – CIM\_Processor.RequestStateChange() method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed

652 **8.2 CIM\_ProcessorCore.RequestStateChange()**

- 653 Invocation of the CIM\_ProcessorCore.RequestStateChange() method changes the element’s state to the
- 654 value that is specified in the RequestedState parameter.
- 655 Return code values for the RequestStateChange() method shall be as specified in Table 11. Parameters
- 656 of the RequestStateChange() method are specified in Table 12.
- 657 When processor core state management is supported, the RequestStateChange() method shall be
- 658 implemented and shall not return a value of 1 (Not Supported) (see 7.6.2.1).
- 659 Invoking the CIM\_ProcessorCore.RequestStateChange() method multiple times could result in earlier
- 660 requests being overwritten or lost.
- 661 No standard messages are defined for this method.

662 **Table 11 – CIM\_ProcessorCore.RequestStateChange() method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

663

**Table 12 – CIM\_ProcessorCore.RequestStateChange() method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed

**664 8.3 CIM\_HardwareThread.RequestStateChange()**

665 Invocation of the CIM\_HardwareThread.RequestStateChange() method changes the element's state to  
666 the value that is specified in the RequestedState parameter.

667 Return code values for the RequestStateChange() method shall be as specified in Table 13. Parameters  
668 of the RequestStateChange() method are specified in Table 14.

669 When hardware thread state management is supported, the RequestStateChange() method shall be  
670 implemented and shall not return a value of 1 (Not Supported) (see 7.7.2.1).

671 Invoking the CIM\_HardwareThread.RequestStateChange() method multiple times could result in earlier  
672 requests being overwritten or lost.

673 No standard messages are defined for this method.

**674 Table 13 – CIM\_HardwareThread.RequestStateChange() method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

675

**Table 14 – CIM\_HardwareThread.RequestStateChange() method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed



676 **8.4 CIM\_Memory.RequestStateChange()**

677 Invocation of the CIM\_Memory.RequestStateChange( ) method changes the element’s state to the value  
678 that is specified in the RequestedState parameter.

679 Return code values for the RequestStateChange( ) method shall be as specified in Table 15. Parameters  
680 of the RequestStateChange( ) method are specified in Table 16.

681 When memory state management is supported, the RequestStateChange( ) method shall be implemented  
682 and shall not return a value of 1 (Not Supported) (see 7.8.2.1).

683 Invoking the CIM\_Memory.RequestStateChange( ) method multiple times could result in earlier requests  
684 being overwritten or lost.

685 No standard messages are defined for this method.

686 **Table 15 – CIM\_Memory.RequestStateChange( ) method: Return code values**

Value	Description
0	Request was successfully executed.
1	Method is not supported in the implementation.
2	Error occurred
4096	Job started

687 **Table 16 – CIM\_Memory.RequestStateChange( ) method: Parameters**

Qualifiers	Name	Type	Description/Values
IN, REQ	RequestedState	uint16	Valid state values: 2 (Enabled) 3 (Disabled) 11 (Reset)
OUT	Job	CIM_ConcreteJob REF	Returned if job started
IN, REQ	TimeoutPeriod	datetime	Client-specified maximum amount of time the transition to a new state is supposed to take: 0 or NULL – No time requirements <interval> – Maximum time allowed

688 **8.5 Profile conventions for operations**

689 This profile specification defines operations in terms of [DSP0200](#).

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

692 The default list of operations is as follows:

- 693 • Associators( )
- 694 • AssociatorNames( )
- 695 • EnumerateInstances( )

- 696 • EnumerateInstanceNames( )
- 697 • GetInstance( )
- 698 • References( )
- 699 • ReferenceNames( )

## 700 8.6 CIM\_AssociatedCacheMemory

701 Table 17 lists implementation requirements for operations. If implemented, these operations shall be  
 702 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 17, all operations  
 703 in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

704 NOTE Related profiles may define additional requirements on operations for the profile class.

705 **Table 17 – Operations: CIM\_AssociatedCacheMemory**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

## 706 8.7 CIM\_ConcreteComponent — References CIM\_HardwareThread and 707 CIM\_Processor

708 Table 18 lists implementation requirements for operations. If implemented, these operations shall be  
 709 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 18, all operations  
 710 in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

711 NOTE Related profiles may define additional requirements on operations for the profile class.

712 **Table 18 – Operations: CIM\_ConcreteComponent**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

## 713 8.8 CIM\_ConcreteComponent — References CIM\_ProcessorCore and 714 CIM\_Processor

715 Table 19 lists implementation requirements for operations. If implemented, these operations shall be  
 716 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 19, all operations  
 717 in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

718 NOTE Related profiles may define additional requirements on operations for the profile class.

719

**Table 19 – Operations: CIM\_ConcreteComponent**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

720

**8.9 CIM\_ElementCapabilities — References CIM\_HardwareThread and CIM\_EnabledLogicalElementCapabilities**

721

722

Table 20 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 20, all operations in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

723

724

725

NOTE Related profiles may define additional requirements on operations for the profile class.

726

**Table 20 – Operations: CIM\_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

727

**8.10 CIM\_ElementCapabilities — References CIM\_Memory and CIM\_EnabledLogicalElementCapabilities**

728

729

Table 21 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 21, all operations in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

730

731

732

NOTE Related profiles may define additional requirements on operations for the profile class.

733

**Table 21 – Operations: CIM\_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

734

**8.11 CIM\_ElementCapabilities — References CIM\_Processor and CIM\_ProcessorCapabilities**

735

736

Table 22 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 22, all operations in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

737

738

739

NOTE Related profiles may define additional requirements on operations for the profile class.

740

**Table 22 – Operations: CIM\_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

741

## 8.12 CIM\_ElementCapabilities — References CIM\_ProcessorCore and CIM\_EnabledLogicalElementCapabilities

742

743

Table 23 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 23, all operations in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

744

745

746

NOTE Related profiles may define additional requirements on operations for the profile class.

747

**Table 23 – Operations: CIM\_ElementCapabilities**

Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

748

## 8.13 CIM\_EnabledLogicalElementCapabilities

749

All operations in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

750

NOTE Related profiles may define additional requirements on operations for the profile class.

751

## 8.14 CIM\_HardwareThread

752

Table 24 lists implementation requirements for operations. If implemented, these operations shall be implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 24, all operations in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

753

754

755

NOTE Related profiles may define additional requirements on operations for the profile class.

756

**Table 24 – Operations: CIM\_HardwareThread**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.14.1.	None

757

### 8.14.1 CIM\_HardwareThread — ModifyInstance

758

This clause details the requirements for the ModifyInstance operation applied to an instance of CIM\_HardwareThread. The ModifyInstance operation may be supported.

759

760

The ModifyInstance operation shall be supported and the CIM\_HardwareThread.ElementName property shall be modifiable when the ElementNameEditSupported property of the

761

762 CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_HardwareThread  
 763 instance has a value of TRUE. See 7.7.1.2.

764 **8.15 CIM\_Memory**

765 Table 25 lists implementation requirements for operations. If implemented, these operations shall be  
 766 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 25, all operations  
 767 in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

768 NOTE Related profiles may define additional requirements on operations for the profile class.

769 **Table 25 – Operations: CIM\_Memory**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.15.1.	None

770 **8.15.1 CIM\_Memory — ModifyInstance**

771 This clause details the requirements for the ModifyInstance operation applied to an instance of  
 772 CIM\_Memory. The ModifyInstance operation may be supported.

773 The ModifyInstance operation shall be supported and the CIM\_Memory.ElementName property shall be  
 774 modifiable when the ElementNameEditSupported property of the  
 775 CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_Memory instance has  
 776 a value of TRUE. See clause 7.8.1.2.

777 **8.16 CIM\_Processor**

778 Table 26 lists implementation requirements for operations. If implemented, these operations shall be  
 779 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 26, all operations  
 780 in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

781 NOTE Related profiles may define additional requirements on operations for the profile class.

782 **Table 26 – Operations: CIM\_Processor**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.16.1.	None

783 **8.16.1 CIM\_Processor — ModifyInstance**

784 This clause details the requirements for the ModifyInstance operation applied to an instance of  
 785 CIM\_Processor. The ModifyInstance operation may be supported.

786 The ModifyInstance operation shall be supported and the CIM\_Processor.ElementName property shall be  
 787 modifiable when the ElementNameEditSupported property of the  
 788 CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_Processor instance  
 789 has a value of TRUE. See 7.2.4.

790 **8.17 CIM\_ProcessorCapabilities**

791 All operations in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

792 NOTE Related profiles may define additional requirements on operations for the profile class.

793 **8.18 CIM\_ProcessorCore**

794 Table 27 lists implementation requirements for operations. If implemented, these operations shall be  
 795 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 27, all operations  
 796 in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

797 NOTE Related profiles may define additional requirements on operations for the profile class.

798 **Table 27 – Operations: CIM\_ProcessorCore**

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.18.1.	None

799 **8.18.1 CIM\_ProcessorCore — ModifyInstance**

800 This clause details the requirements for the ModifyInstance operation applied to an instance of  
 801 CIM\_ProcessorCore. The ModifyInstance operation may be supported.

802 The ModifyInstance operation shall be supported and the CIM\_ProcessorCore.ElementName property  
 803 shall be modifiable when the ElementNameEditSupported property of the  
 804 CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_ProcessorCore  
 805 instance has a value of TRUE. See 7.6.1.2.

806 **8.19 CIM\_SystemDevice**

807 Table 28 lists implementation requirements for operations. If implemented, these operations shall be  
 808 implemented as defined in [DSP0200](#). In addition, and unless otherwise stated in Table 28, all operations  
 809 in the default list in 8.5 shall be implemented as defined in [DSP0200](#).

810 NOTE Related profiles may define additional requirements on operations for the profile class.

811 **Table 28 – Operations: CIM\_SystemDevice**

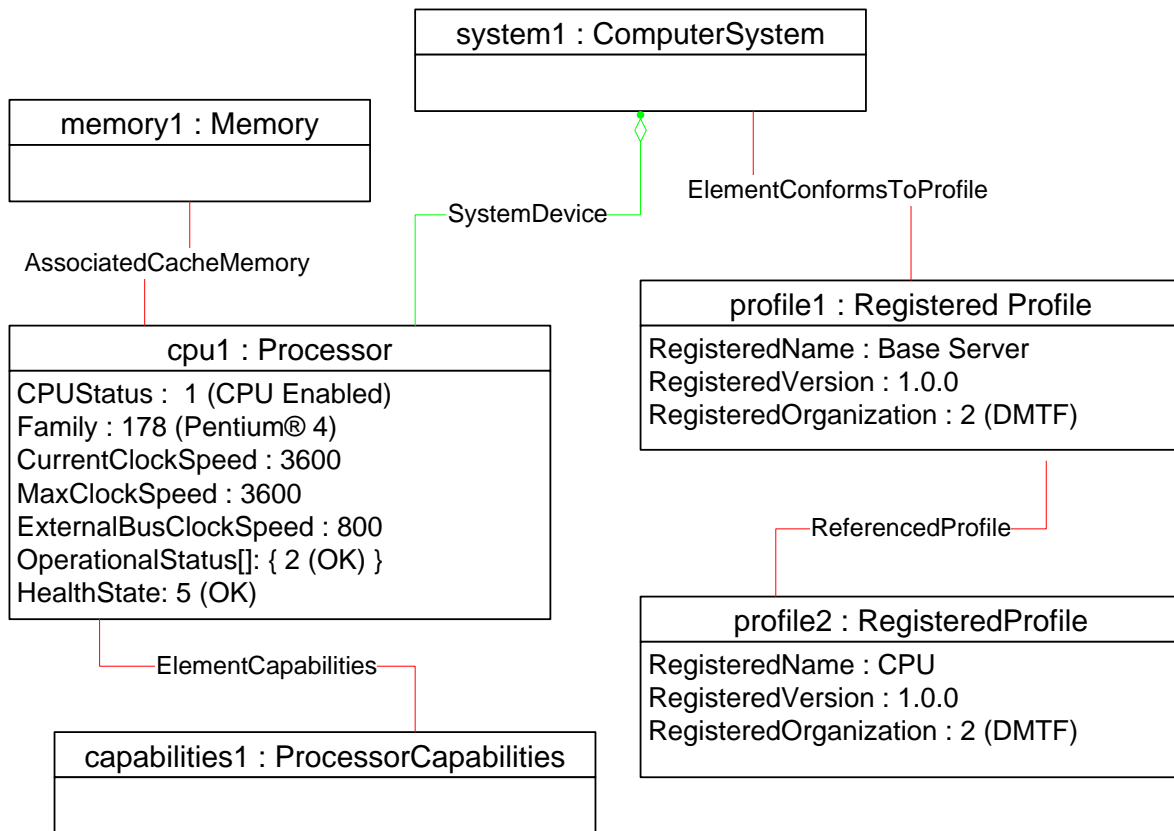
Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

812 **9 Use cases**

813 This clause contains object diagrams and use cases for the *CPU Profile*.

814 **9.1 Object diagrams**

815 Figure 2 represents a possible instantiation of the *CPU Profile*. In this instantiation, cpu1 belongs to  
 816 system1. The capabilities of cpu1 are represented with capabilities1. cpu1 has cache memory  
 817 represented by memory1. The *CPU Profile* implementation and versioning information is advertised  
 818 through profile2.

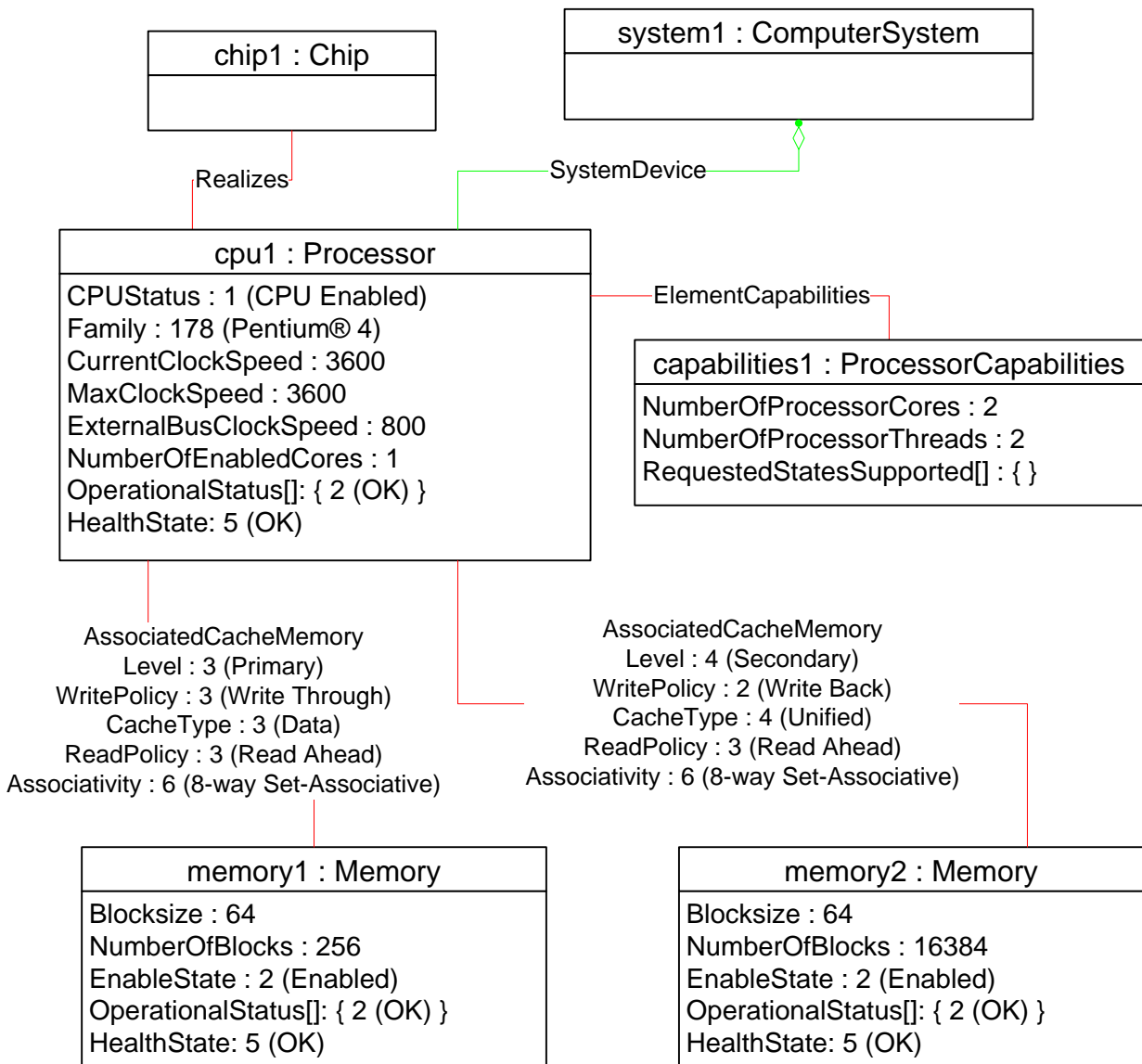


819

820

**Figure 2 – Registered Profile**

821 Figure 3 represents a possible instantiation of the *CPU Profile* representing a dual core processor, cpu1.  
 822 The individual cores and hardware threads of cpu1 are not represented, but capabilities1 advertises that  
 823 cpu1 is a dual core processor capable of two hardware threads, one thread per each core. If system1  
 824 supports [SMBIOS Reference Specification 2.6](#) or later, the value of the NumberOfProcessorCores  
 825 property will be equal to the SMBIOS Processor Information structure's Core Count structure value, and  
 826 the value of the NumberOfHardwareThreads property will be equal to the SMBIOS Processor Information  
 827 structure's Thread Count structure value. memory1 and memory2 are the cache memories of cpu1.  
 828 Memory1 represents the level 1 cache, and memory2 is the level 2 cache, as denoted by the instances of  
 829 CIM\_AssociatedCacheMemory that associate memory1 and memory2 with cpu1. The physical aspects of  
 830 cpu1 are represented by chip1, associated to cpu1 through an instance of CIM\_Realizes.



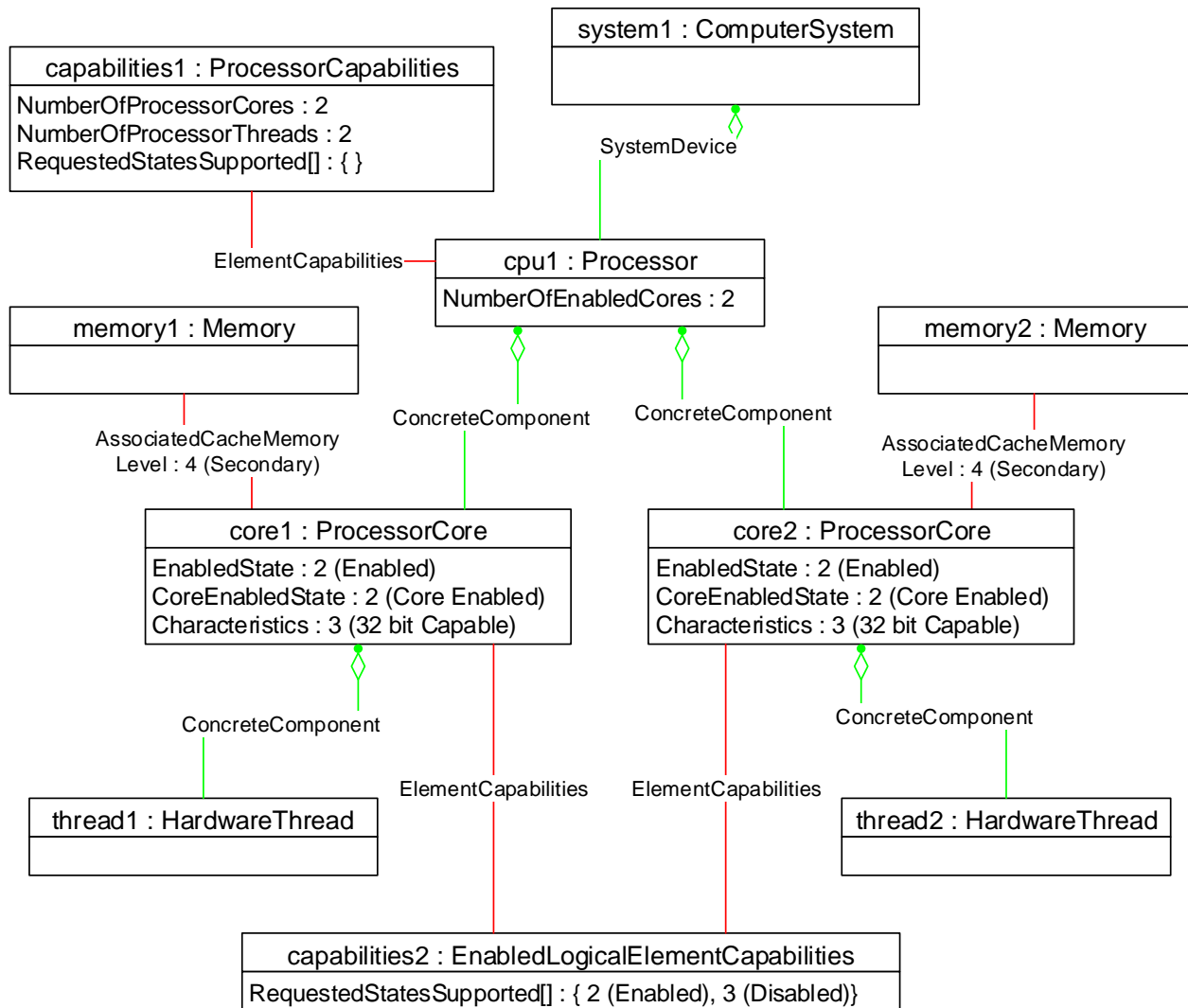
831

832

Figure 3 – Multi-Core CPU



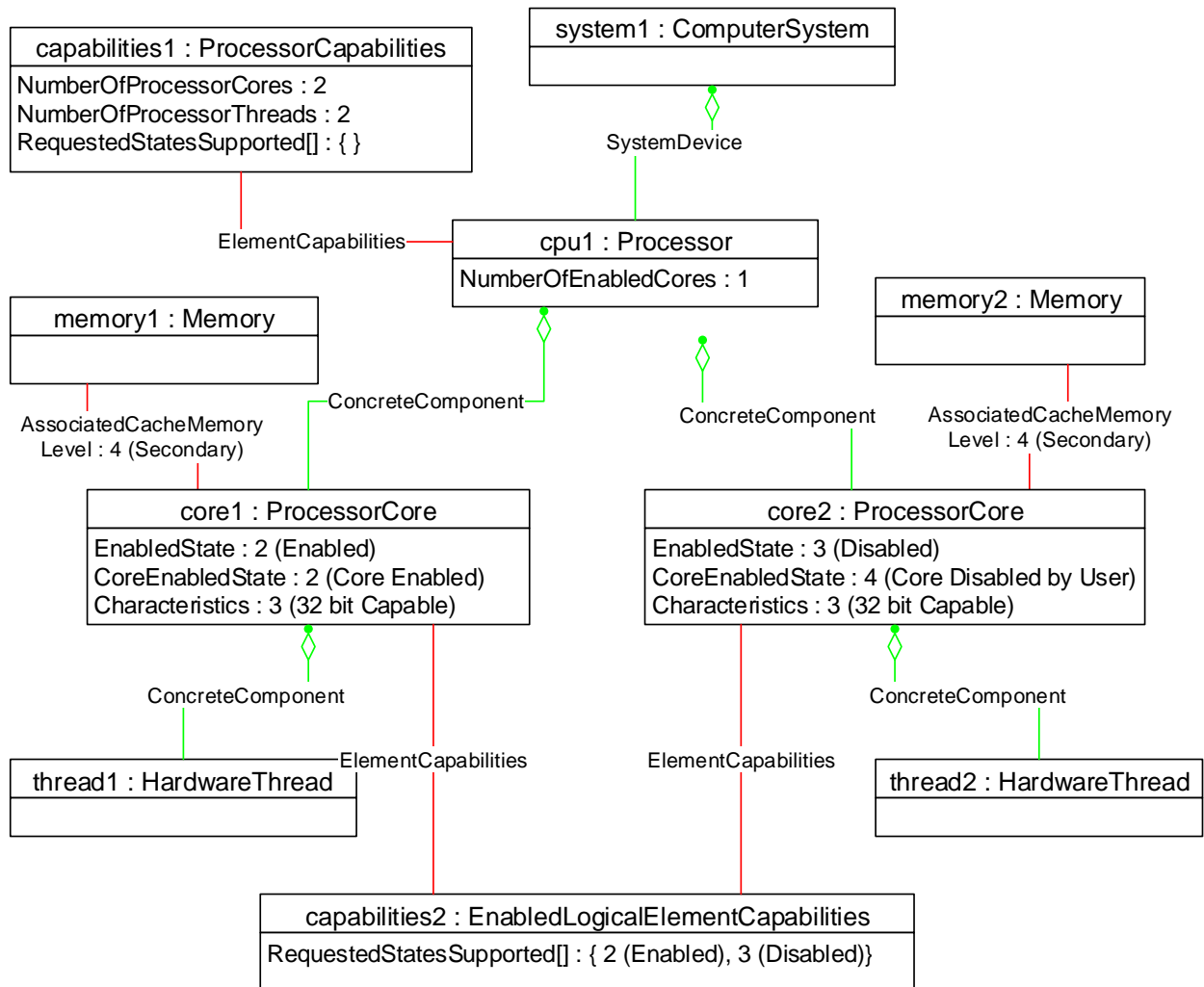
833 Figure 4 represents a possible instantiation of the *CPU Profile* representing a dual core processor, cpu1.  
 834 Each of the processor cores is represented by an instance of CIM\_ProcessorCore: core1 and core2,  
 835 associated to the Host Processor, cpu1, through instances of CIM\_ConcreteComponent. Each of the  
 836 cores has one hardware thread, represented by thread1 and thread2, associated with it through instances  
 837 of CIM\_ConcreteComponent. The cache memories, memory1 and memory2, are associated to the  
 838 processor cores that use them. Based on the capabilities of core1 and core2, represented by  
 839 capabilities2, both processor cores can be enabled or disabled using the RequestStateChange() method.  
 840 Figure 5 shows the same instantiation of *CPU Profile* after the RequestStateChange() method on core2  
 841 has successfully executed.



**Figure 4 – Detailed Multi-Core CPU**

844 Figure 5 represents a possible instantiation of the *CPU Profile* in which one of the cores of a dual core  
 845 processor, cpu1, has been disabled by the user using the RequestStateChange() method. core2's  
 846 EnabledState property has value of 3 (Disabled) and the CoreEnabledState property has value 4 (Core  
 847 Disabled by User).

848



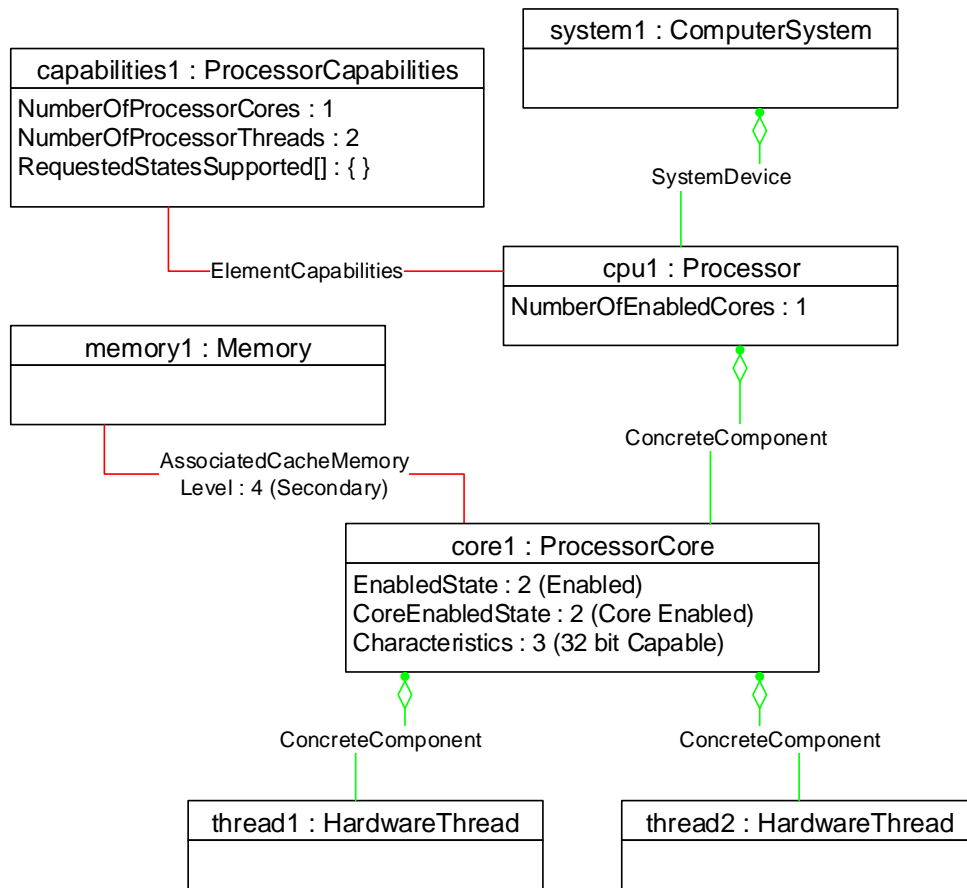
849

850

**Figure 5 – Multi-Core CPU with a Disabled Core**

851 Figure 6 represents a possible instantiation of the *CPU Profile* representing a single core processor with  
 852 multiple threads. thread1 and thread2 represent the hardware threads that exist on core1 and are  
 853 associated to core1 through an instance of CIM\_ConcreteComponent. cpu1 advertises the capabilities of  
 854 multiple hardware threads through the capabilities1 NumberOfProcessorThreads property. The cache  
 855 memory, memory1, is associated to core1, which is using the cache memory.

856

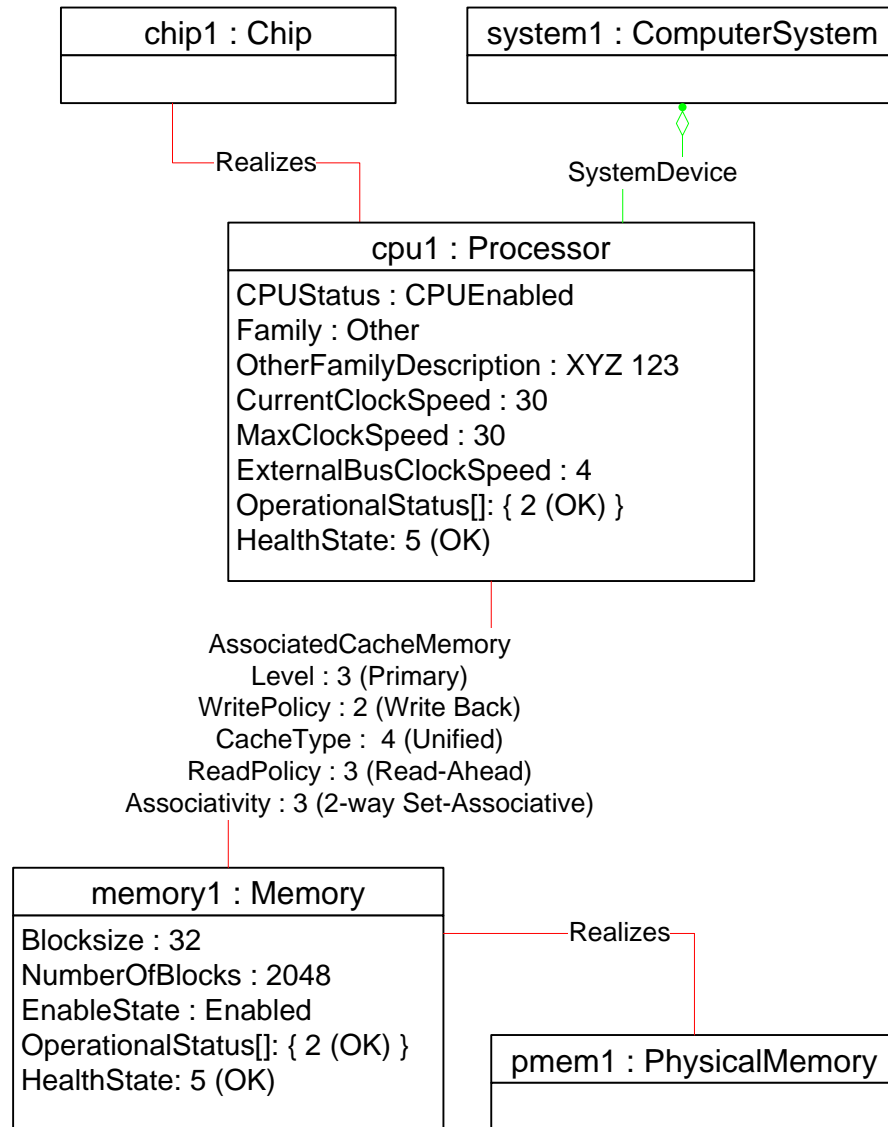


857

858

**Figure 6 – Single Core, Multi-Hardware Thread CPU**

859 Figure 7 represents another instantiation of the *CPU Profile*. In this case, cpu1’s cache memory,  
 860 memory1, has a separate package represented by pmem1 and associated to memory1 through an  
 861 instance of CIM\_Realizes. The existence of pmem1 associated with the cpu1’s cache memory indicates  
 862 that the processor uses off-chip cache memory.



863

864

**Figure 7 – Processor with Off-Chip Cache**

## 865 9.2 Change the enabled state of the memory to the desired state

866 A client can change the enabled state of the memory as follows:

- 867 1) Select the instance of CIM\_Memory.
- 868 2) Select the associated instance of CIM\_EnabledLogicalElementCapabilities and verify whether  
869 the RequestedStatesSupported property contains the desired state.
- 870 3) If the RequestedStatesSupported property contains the desired state, select the instance of  
871 CIM\_Memory and execute the RequestStateChange() method with the desired state as a  
872 RequestedState parameter.

873 After the successful execution of the method, the EnabledState property of the instance of CIM\_Memory  
874 will have the value of the desired state.

### 875 **9.3 Change the enabled state of the CPU to the desired state**

876 A client can change the enabled state of the CPU as follows:

- 877 1) Select the instance of CIM\_Processor.
- 878 2) Select the associated instance of CIM\_ProcessorCapabilities and verify whether the  
879 RequestedStatesSupported property contains the desired state.
- 880 3) If the RequestedStatesSupported property contains the desired state, select the instance of  
881 CIM\_Processor and execute the RequestStateChange( ) method with the desired state as a  
882 RequestedState parameter.

883 After the successful execution of the method, the EnabledState property of the instance of  
884 CIM\_Processor will have the value of the desired state.

### 885 **9.4 Change the enabled state of the CPU's core to the desired state**

886 A client can change the enabled state of the CPU's core as follows:

- 887 1) Select the instance of CIM\_ProcessorCore.
- 888 2) Select the associated instance of CIM\_EnabledLogicalElementCapabilities and verify whether  
889 the RequestedStatesSupported property contains the desired state.
- 890 3) If the RequestedStatesSupported property contains the desired state, select the instance of  
891 CIM\_ProcessorCore and execute the RequestStateChange( ) method with the desired state as  
892 a RequestedState parameter.

893 After the successful execution of the method, the EnabledState property of the instance of  
894 CIM\_ProcessorCore will have the value of the desired state.

### 895 **9.5 Change the enabled state of the CPU's hardware thread to the desired state**

896 A client can change the enabled state of the CPU's hardware thread as follows:

- 897 1) Select the instance of CIM\_HardwareThread.
- 898 2) Select the associated instance of CIM\_EnabledLogicalElementCapabilities and verify whether  
899 the RequestedStatesSupported property contains the desired state.
- 900 3) If the RequestedStatesSupported property contains the desired state, select the instance of  
901 CIM\_ProcessorThread and execute the RequestStateChange( ) method with the desired state  
902 as a RequestedState parameter.

903 After the successful execution of the method, the EnabledState property of the instance of  
904 CIM\_HardwareThread will have the value of the desired state.

### 905 **9.6 Retrieve all the processor cores for the CPU**

906 A client can retrieve all of the processor cores for the CPU by selecting all the CIM\_ProcessorCore  
907 instances that are associated with the given instance of CIM\_Processor through instances of  
908 CIM\_Component.

909 **9.7 Retrieve all the hardware threads for the CPU**

910 A client can retrieve all of the hardware threads for the CPU as follows:

- 911 1) Select all the CIM\_ProcessorCore instances that are associated with the given instance of
- 912 CIM\_Processor through instances of CIM\_Component.
- 913 2) For each instance of CIM\_ProcessorCore, select the instances of CIM\_HardwareThread that
- 914 are associated through instances of CIM\_Component.

915 **9.8 Retrieve CPU’s cache memory information for the CPU**

916 A client can retrieve the CPU’s cache memory information as follows:

- 917 1) Select all the instances of CIM\_ProcessorCore that are associated with the given instance of
- 918 CIM\_Processor through instances of CIM\_Component.
- 919 2) If no instance of CIM\_ProcessorCore exists, select the instances of
- 920 CIM\_AssociatedCacheMemory that reference the given instance of CIM\_Processor, as well as
- 921 all the instances of CIM\_Memory that are associated with the given instance of CIM\_Processor
- 922 through instances of CIM\_AssociatedCacheMemory.
- 923 3) Otherwise, for each instance of CIM\_ProcessorCore, select the instances of
- 924 CIM\_AssociatedCacheMemory that reference the instance of CIM\_ProcessorCore, as well as
- 925 all the instances of CIM\_Memory that are associated with the instance of CIM\_ProcessorCore
- 926 through instances of CIM\_AssociatedCacheMemory.

927 **10 CIM Elements**

928 Table 29 shows the instances of CIM Elements for this profile. Instances of the CIM Elements shall be  
 929 implemented as described in Table 29. Clauses 7 (“Implementation”) and 8 (“Methods”) may impose  
 930 additional requirements on these elements.

931 **Table 29 – CIM Elements: CPU Profile**

Element Name	Requirement	Description
<b>Classes</b>		
CIM_AssociatedCacheMemory	Optional	See 10.1 and 7.8.
CIM_ConcreteComponent (references CIM_HardwareThread and CIM_ProcessorCore)	Optional	See 10.2.
CIM_ConcreteComponent (references CIM_ProcessorCore and CIM_Processor)	Optional	See 10.3.
CIM_ElementCapabilities (references CIM_HardwareThread and CIM_EnabledLogicalElementCapabilities)	Optional	See 10.4.
CIM_ElementCapabilities (references CIM_Memory and CIM_EnabledLogicalElementCapabilities)	Optional	See 10.5.
CIM_ElementCapabilities (references CIM_Processor and CIM_ProcessorCapabilities)	Optional	See 10.6.

Element Name	Requirement	Description
CIM_ElementCapabilities (references CIM_ProcessorCore and CIM_EnabledLogicalElementCapabilities)	Optional	See 10.7.
CIM_EnabledLogicalElementCapabilities	Optional	See 7.6.1, 7.7.1, 7.8.1, and 10.7.
CIM_HardwareThread	Optional	See 10.9.
CIM_Memory	Optional	See 10.10 and 7.8.
CIM_Processor	Mandatory	See 7.1 and 10.11.
CIM_ProcessorCapabilities	Optional	See 7.2 and 10.12.
CIM_ProcessorCore	Optional	See 10.13.
CIM_RegisteredProfile	Mandatory	See 10.14.
CIM_SystemDevice	Mandatory	See 10.15.
<b>Indications</b>		
None defined in this profile		

932 **10.1 CIM\_AssociatedCacheMemory**

933 CIM\_AssociatedCacheMemory associates an instance of CIM\_Processor or CIM\_ProcessorCore with an  
 934 instance of CIM\_Memory that represents the cache memory of the processor. Table 30 contains the  
 935 requirements for elements of this class.

936 **Table 30 – Class: CIM\_AssociatedCacheMemory**

Elements	Requirement	Notes
Antecedent	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_Memory that represents the cache memory.
Dependent	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_Processor or CIM_ProcessorCore. See 7.8 for more details.
Level	Mandatory	None
WritePolicy	Mandatory	None
CacheType	Mandatory	None
ReadPolicy	Mandatory	None
Associativity	Mandatory	None
OtherLevelDescription	Conditional	This property shall be implemented when the Level property has a value of 1 (Other).
OtherWritePolicyDescription	Conditional	This property shall be implemented when the WritePolicy property has a value of 1 (Other).
OtherCacheTypeDescription	Conditional	This property shall be implemented when the CacheType property has a value of 1 (Other).

937 **10.2 CIM\_ConcreteComponent — References CIM\_HardwareThread and**  
 938 **CIM\_ProcessorCore**

939 CIM\_ConcreteComponent associates an instance of CIM\_ProcessorCore (the Threading Processor Core)  
 940 with an instance CIM\_HardwareThread that represents a hardware thread. CIM\_ConcreteComponent  
 941 shall be instantiated when the Threading Processor Core and the instance of CIM\_HardwareThread are  
 942 instantiated. Table 31 contains the requirements for elements of this class.

943 **Table 31 – Class: CIM\_ConcreteComponent — References CIM\_HardwareThread and**  
 944 **CIM\_ProcessorCore**

Elements	Requirement	Notes
GroupComponent	Mandatory	<b>Key:</b> This property shall reference the Threading Processor Core.
PartComponent	Mandatory	<b>Key:</b> This property shall reference the CIM_HardwareThread that represents the hardware thread.

945 **10.3 CIM\_ConcreteComponent — References CIM\_ProcessorCore and**  
 946 **CIM\_Processor**

947 CIM\_ConcreteComponent associates an instance of CIM\_Processor (the Host Processor) with an  
 948 instance CIM\_ProcessorCore that represents a processor core. CIM\_ConcreteComponent shall be  
 949 instantiated when the Host Processor and the instance of CIM\_ProcessorCore are instantiated. Table 32  
 950 contains the requirements for elements of this class.

951 **Table 32 – Class: CIM\_ConcreteComponent — References CIM\_ProcessorCore and**  
 952 **CIM\_Processor**

Elements	Requirement	Notes
GroupComponent	Mandatory	<b>Key:</b> This property shall reference the Host Processor.
PartComponent	Mandatory	<b>Key:</b> This property shall reference the CIM_ProcessorCore that represents the hosted processor cores.

953 **10.4 CIM\_ElementCapabilities — References CIM\_HardwareThread and**  
 954 **CIM\_EnabledLogicalElementCapabilities**

955 CIM\_ElementCapabilities associates an instance of CIM\_HardwareThread with the instance of  
 956 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of  
 957 CIM\_HardwareThread.

958 CIM\_ElementCapabilities is mandatory when the instance of CIM\_HardwareThread and the instance of  
 959 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of  
 960 CIM\_HardwareThread exist. Table 33 contains the requirements for elements of this class.



961 **Table 33 – Class: CIM\_ElementCapabilities — References CIM\_HardwareThread and**  
 962 **CIM\_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_HardwareThread.
Capabilities	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_EnabledLogicalElementCapabilities.

963 **10.5 CIM\_ElementCapabilities — References CIM\_Memory and**  
 964 **CIM\_EnabledLogicalElementCapabilities**

965 CIM\_ElementCapabilities associates an instance of CIM\_Memory with the instance of  
 966 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of CIM\_Memory.

967 CIM\_ElementCapabilities is mandatory when the instance of CIM\_Memory and the instance of  
 968 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of CIM\_Memory  
 969 exist. Table 34 contains the requirements for elements of this class.

970 **Table 34 – Class: CIM\_ElementCapabilities — References CIM\_Memory and**  
 971 **CIM\_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_Memory.
Capabilities	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_EnabledLogicalElementCapabilities.

972 **10.6 CIM\_ElementCapabilities — References CIM\_Processor and**  
 973 **CIM\_ProcessorCapabilities**

974 CIM\_ElementCapabilities associates an instance of CIM\_Processor with the instance of  
 975 CIM\_ProcessorCapabilities that describes the capabilities of the instance of CIM\_Processor.

976 CIM\_ElementCapabilities is mandatory when the instance of CIM\_Processor and the instance of  
 977 CIM\_ProcessorCapabilities exist. Table 35 contains the requirements for elements of this class.

978 **Table 35 – Class: CIM\_ElementCapabilities — References CIM\_Processor and**  
 979 **CIM\_ProcessorCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_Processor.
Capabilities	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_ProcessorCapabilities.

980 **10.7 CIM\_ElementCapabilities — References CIM\_ProcessorCore and**  
 981 **CIM\_EnabledLogicalElementCapabilities**

982 CIM\_ElementCapabilities associates an instance of CIM\_ProcessorCore with the instance of  
 983 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of  
 984 CIM\_ProcessorCore.

985 CIM\_ElementCapabilities is mandatory when the instance of CIM\_ProcessorCore and the instance of  
 986 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of  
 987 CIM\_ProcessorCore exist. Table 36 contains the requirements for elements of this class.

988 **Table 36 – Class: CIM\_ElementCapabilities — References CIM\_ProcessorCore and**  
 989 **CIM\_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
ManagedElement	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_ProcessorCore.
Capabilities	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_EnabledLogicalElementCapabilities.

## 990 10.8 CIM\_EnabledLogicalElementCapabilities

991 CIM\_EnabledLogicalElementCapabilities represents the capabilities of the memory, the processor core,  
 992 or the hardware thread. Table 37 contains the requirements for elements of this class.

993 **Table 37 – Class: CIM\_EnabledLogicalElementCapabilities**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
RequestedStatesSupported	Mandatory	See 7.6.1.1, 7.7.1.1, and 7.8.1.1.
ElementNameEditSupported	Mandatory	See 7.6.1.2, 7.7.1.2, and 7.8.1.1.
MaxElementNameLen	Conditional	See 7.6.1.3, 7.7.1.3, and 7.8.1.3.

## 994 10.9 CIM\_HardwareThread

995 CIM\_HardwareThread represents the hardware thread of the processor. Table 38 contains the  
 996 requirements for elements of this class.

997 **Table 38 – Class: CIM\_HardwareThread**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
EnabledState	Mandatory	See 7.7.4.
RequestedState	Mandatory	See 7.7.3.
OperationalStatus	Mandatory	None
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern “.*”.
RequestStateChange()	Conditional	See 8.3.

## 998 10.10 CIM\_Memory

999 CIM\_Memory represents the CPU’s cache memory. Table 39 contains the requirements for elements of  
 1000 this class.

1001

**Table 39 – Class: CIM\_Memory**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	<b>Key</b>
CreationClassName	Mandatory	<b>Key</b>
SystemName	Mandatory	<b>Key</b>
DeviceID	Mandatory	<b>Key</b>
BlockSize	Mandatory	None
NumberOfBlocks	Mandatory	None
EnabledState	Mandatory	See 7.8.4.
RequestedState	Mandatory	See 7.8.3.
HealthState	Mandatory	None
OperationalStatus	Mandatory	None
ElementName	Mandatory	The property shall match the pattern “.*”.
RequestStateChange()	Conditional	See 8.4.

1002 **10.11 CIM\_Processor**

1003 CIM\_Processor represents the processor or CPU. Table 40 contains the requirements for elements of this  
 1004 class.

1005

**Table 40 – Class: CIM\_Processor**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	<b>Key</b>
SystemName	Mandatory	<b>Key</b>
CreationClassName	Mandatory	<b>Key</b>
DeviceID	Mandatory	<b>Key</b>
Family	Mandatory	None
CurrentClockSpeed	Mandatory	When the EnabledState property has a value of 2 (Enabled), a value of 0 shall indicate that the property value is unknown. When the EnabledState property has a value of 3 (Disabled), this property shall have no meaning.
MaxClockSpeed	Mandatory	When the EnabledState property has a value of 2 (Enabled), a value of 0 shall indicate that the property value is unknown. When the EnabledState property has a value of 3 (Disabled), this property shall have no meaning.
ExternalBusClockSpeed	Mandatory	When the EnabledState property has a value of 2 (Enabled), a value of 0 shall indicate that the property value is unknown. When the EnabledState property has a value of 3 (Disabled), this property shall have no meaning.
CPUStatus	Mandatory	See 7.5.1.
EnabledState	Mandatory	See 7.5.2.
RequestedState	Mandatory	See 7.4.
OperationalStatus	Mandatory	None

Elements	Requirement	Notes
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern “.*”.
OtherFamilyDescription	Conditional	This property shall be implemented if the Family property contains the value “Other”.
RequestStateChange()	Conditional	See 8.1.

## 1006 10.12 CIM\_ProcessorCapabilities

1007 CIM\_ProcessorCapabilities represents the capabilities of the processor. Table 41 contains the  
1008 requirements for elements of this class.

1009 **Table 41 – Class: CIM\_ProcessorCapabilities**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
NumberOfProcessorCores	Mandatory	A value of 0 shall mean “Unknown”.
NumberOfHardwareThreads	Mandatory	A value of 0 shall mean “Unknown”.
RequestedStatesSupported	Mandatory	See 7.2.2.
ElementNameEditSupported	Mandatory	See 7.2.4.
MaxElementNameLen	Conditional	See 7.2.5.
ProcessorArchitecture	Optional	EXPERIMENTAL
InstructionSet	Optional	EXPERIMENTAL
InstructionSetExtensionName	Optional	EXPERIMENTAL
InstructionSetExtensionStatus	Optional	EXPERIMENTAL

## 1010 10.13 CIM\_ProcessorCore

1011 CIM\_ProcessorCore represents the core of the processor. Table 42 contains the requirements for  
1012 elements of this class.

1013 **Table 42 – Class: CIM\_ProcessorCore**

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
CoreEnabledState	Mandatory	See 7.6.4.1.
EnabledState	Mandatory	See 7.6.4.2.
RequestedState	Mandatory	See 7.6.3.
OperationalStatus	Mandatory	None
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern “.*”.
RequestStateChange()	Conditional	See 8.2.

1014 **10.14 CIM\_RegisteredProfile**

1015 The CIM\_RegisteredProfile class is defined by the [Profile Registration Profile](#). The requirements denoted  
 1016 in Table 43 are in addition to those mandated by the [Profile Registration Profile](#).

1017 **Table 43 – Class: CIM\_RegisteredProfile**

Elements	Requirement	Notes
RegisteredName	Mandatory	This property shall have a value of “CPU”.
RegisteredVersion	Mandatory	This property shall have a value of “1.1.0”.
RegisteredOrganization	Mandatory	This property shall have a value of 2 (DMTF).

1018 NOTE Previous versions of this document included the suffix “Profile” for the RegisteredName value. If  
 1019 implementations querying for the RegisteredName value find the suffix “Profile”, they should ignore the suffix, with  
 1020 any surrounding white spaces, before any comparison is done with the value as specified in this document.

1021 **10.15 CIM\_SystemDevice**

1022 CIM\_SystemDevice associates an instance of CIM\_Processor with the instance of CIM\_ComputerSystem  
 1023 of which the CIM\_Processor instance is a member. Table 44 contains the requirements for elements of  
 1024 this class.

1025 **Table 44 – Class: CIM\_SystemDevice**

Elements	Requirement	Notes
GroupComponent	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_ComputerSystem of which the instance of CIM_Processor is a member.
PartComponent	Mandatory	<b>Key:</b> This property shall reference the instance of CIM_Processor.

1026

**ANNEX A  
(informative)****Change log**

Version	Date	Description
1.0.0	2008-10-31	Final Version of the Profile
1.0.1	2010-04-22	Released as DMTF Standard — Changed ExternalClockSpeed to ExternalBusClockSpeed in use cases to be in sync with the MOF
1.0.2	2015-05-04	Released as DMTF Standard
1.1.0	2016-04-14	Added experimental properties from CIM_ProcessorCapabilities

1027  
1028  
1029  
10301031  
1032