



1

2

3

4

**Document Number: DSP1022**

**Date: 2010-04-22**

**Version: 1.0.1**

5 **CPU Profile**

6 **Document Type: Specification**

7 **Document Status: DMTF Standard**

8 **Document Language: en-US**

9

## 10 Copyright Notice

11 Copyright © 2008–2010 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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

# CONTENTS

34	Foreword .....	7
35	Introduction .....	8
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 .....	13
46	7.4 CIM_Processor.RequestedState .....	13
47	7.5 Modeling the Current Enabled State of the Processor .....	14
48	7.6 Modeling Individual Processor Cores .....	14
49	7.7 Modeling Individual Hardware Threads .....	17
50	7.8 Modeling Cache Memory .....	19
51	7.9 Modeling Physical Aspects of Processor and Cache Memory .....	21
52	8 Methods.....	21
53	8.1 CIM_Processor.RequestStateChange( ) .....	21
54	8.2 CIM_ProcessorCore.RequestStateChange( ) .....	22
55	8.3 CIM_HardwareThread.RequestStateChange( ) .....	23
56	8.4 CIM_Memory.RequestStateChange( ) .....	24
57	8.5 Profile Conventions for Operations.....	24
58	8.6 CIM_AssociatedCacheMemory .....	25
59	8.7 CIM_ConcreteComponent — References CIM_HardwareThread and CIM_Processor .....	25
60	8.8 CIM_ConcreteComponent — References CIM_ProcessorCore and CIM_Processor .....	25
61	8.9 CIM_ElementCapabilities — References CIM_HardwareThread and CIM_EnabledLogicalElementCapabilities.....	26
62	8.10 CIM_ElementCapabilities — References CIM_Memory and CIM_EnabledLogicalElementCapabilities.....	26
63	8.11 CIM_ElementCapabilities — References CIM_Processor and CIM_ProcessorCapabilities .....	26
64	8.12 CIM_ElementCapabilities — References CIM_ProcessorCore and CIM_EnabledLogicalElementCapabilities.....	27
65	8.13 CIM_EnabledLogicalElementCapabilities.....	27
66	8.14 CIM_HardwareThread .....	27
67	8.15 CIM_Memory .....	28
68	8.16 CIM_Processor .....	28
69	8.17 CIM_ProcessorCapabilities .....	28
70	8.18 CIM_ProcessorCore .....	29
71	8.19 CIM_SystemDevice .....	29
72	9 Use Cases.....	30
73	9.1 Object Diagrams .....	30
74	9.2 Change the Enabled State of the Memory to the Desired State .....	35
75	9.3 Change the Enabled State of the CPU to the Desired State .....	36
76	9.4 Change the Enabled State of the CPU's Core to the Desired State .....	36
77	9.5 Change the Enabled State of the CPU's Hardware Thread to the Desired State .....	36
78	9.6 Retrieve All the Processor Cores for the CPU.....	37
79	9.7 Retrieve All the Hardware Threads for the CPU.....	37
80	9.8 Retrieve CPU's Cache Memory Information for the CPU.....	37
81	10 CIM Elements.....	37

86	10.1	CIM_AssociatedCacheMemory .....	39
87	10.2	CIM_ConcreteComponent — References CIM_HardwareThread and	
88		CIM_ProcessorCore .....	39
89	10.3	CIM_ConcreteComponent — References CIM_ProcessorCore and CIM_Processor .....	40
90	10.4	CIM_ElementCapabilities — References CIM_HardwareThread and	
91		CIM_EnabledLogicalElementCapabilities.....	40
92	10.5	CIM_ElementCapabilities — References CIM_Memory and	
93		CIM_EnabledLogicalElementCapabilities.....	40
94	10.6	CIM_ElementCapabilities — References CIM_Processor and	
95		CIM_ProcessorCapabilities .....	41
96	10.7	CIM_ElementCapabilities — References CIM_ProcessorCore and	
97		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.....	44
105	10.15	CIM_SystemDevice .....	45
106	ANNEX A (informative)	Change Log.....	46
107			

## 108 Figures

109	Figure 1 – CPU Profile: Class Diagram .....	11
110	Figure 2 – Registered Profile .....	30
111	Figure 3 – Multi-Core CPU.....	31
112	Figure 4 – Detailed Multi-Core CPU .....	32
113	Figure 5 – Multi-core CPU with a Disabled Core .....	33
114	Figure 6 – Single Core, Multi-Hardware Thread CPU .....	34
115	Figure 7 – Processor with Off-Chip Cache .....	35
116		

## 117 Tables

118	Table 1 – Related Profiles.....	10
119	Table 2 – CIM_ProcessorCapabilities Properties Mapping to SMBIOS Equivalence .....	12
120	Table 3 – CIM_Processor.CPUStatus Value Descriptions .....	14
121	Table 4 – Mapping for CPUStatus Property and EnabledState Property Values .....	14
122	Table 5 – CIM_ProcessorCore.CoreEnabledState Value Descriptions.....	16
123	Table 6 – Mapping for the CoreEnabledState Property and EnabledState Property Values .....	17
124	Table 7 – CIM_HardwareThread.EnabledState Value Descriptions .....	19
125	Table 8 – CIM_Memory.EnabledState Value Descriptions.....	21
126	Table 9 – CIM_Processor.RequestStateChange( ) Method: Return Code Values.....	22
127	Table 10 – CIM_Processor.RequestStateChange( ) Method: Parameters .....	22
128	Table 11 – CIM_ProcessorCore.RequestStateChange( ) Method: Return Code Values.....	22
129	Table 12 – CIM_ProcessorCore.RequestStateChange( ) Method: Parameters.....	23
130	Table 13 – CIM_HardwareThread.RequestStateChange( ) Method: Return Code Values.....	23
131	Table 14 – CIM_HardwareThread.RequestStateChange( ) Method: Parameters.....	23
132	Table 15 – CIM_Memory.RequestStateChange( ) Method: Return Code Values.....	24

133 Table 16 – CIM\_Memory.RequestStateChange( ) Method: Parameters..... 24

134 Table 17 – Operations: CIM\_AssociatedCacheMemory..... 25

135 Table 18 – Operations: CIM\_ConcreteComponent ..... 25

136 Table 19 – Operations: CIM\_ConcreteComponent ..... 26

137 Table 20 – Operations: CIM\_ElementCapabilities ..... 26

138 Table 21 – Operations: CIM\_ElementCapabilities ..... 26

139 Table 22 – Operations: CIM\_ElementCapabilities ..... 27

140 Table 23 – Operations: CIM\_ElementCapabilities ..... 27

141 Table 24 – Operations: CIM\_HardwareThread..... 27

142 Table 25 – Operations: CIM\_Memory..... 28

143 Table 26 – Operations: CIM\_Processor..... 28

144 Table 27 – Operations: CIM\_ProcessorCore..... 29

145 Table 28 – Operations: CIM\_SystemDevice..... 29

146 Table 29 – CIM Elements: CPU Profile..... 37

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

148 Table 31 – Class: CIM\_ConcreteComponent — References CIM\_HardwareThread and  
149 CIM\_ProcessorCore..... 39

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 ..... 40

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 ..... 41

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

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

161 Table 39 – Class: CIM\_Memory ..... 42

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..... 44

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

167

168

169

170

## Foreword

171 The *CPU Profile* (DSP1022) was prepared by the Physical Platform Profiles Working Group of the DMTF.

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

### 174 **Acknowledgments**

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

176 Editors:

- 177 • Jon Hass – Dell
- 178 • Khachatur Papanyan – Dell
- 179 • Jeff Hilland – HP

180 Contributors:

- 181 • Jon Hass – Dell
- 182 • Khachatur Papanyan – Dell
- 183 • Jeff Hilland – HP
- 184 • Christina Shaw – HP
- 185 • Aaron Merkin – IBM
- 186 • Jeff Lynch – IBM
- 187 • Perry Vincent – Intel
- 188 • John Leung – Intel

189

190

## Introduction

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

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

## 197 Document Conventions

### 198 Experimental Material

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

204 The following typographical convention indicates experimental material:

---

---

#### 205 **EXPERIMENTAL**

206 Experimental material appears here.

#### 207 **EXPERIMENTAL**

---

---

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

210

211



212

# CPU Profile

## 213 1 Scope

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

## 217 2 Normative References

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

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

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

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

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

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

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

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

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

## 237 3 Terms and Definitions

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

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

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

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

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

### 253 3.1

#### 254 **Cache Memory**

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

### 256 3.2

#### 257 **Host Processor**

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

### 259 3.3

#### 260 **Threading Processor Core**

261 indicates the instance of CIM\_ProcessorCore that represents the processor core that enables the  
 262 hardware threading

## 263 4 Symbols and Abbreviated Terms

### 264 4.1

#### 265 **CPU**

266 central processing unit

## 267 5 Synopsis

268 **Profile Name:** *CPU*

269 **Version:** 1.0.1

270 **Organization:** DMTF

271 **CIM Schema Version:** 2.19

272 **Central Class:** CIM\_Processor

273 **Scoping Class:** CIM\_ComputerSystem

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

276

**Table 1 – Related Profiles**

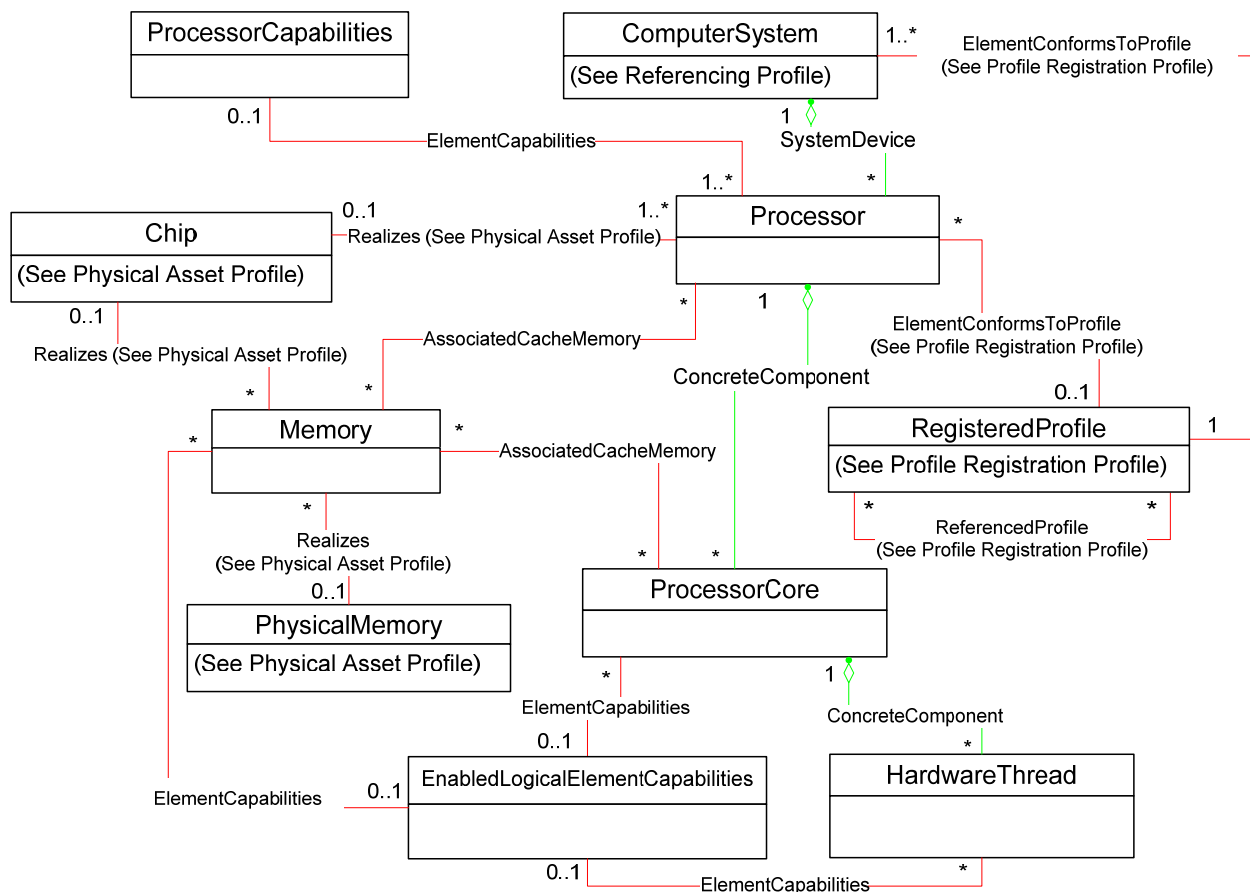
Profile Name	Organization	Version	Requirement	Description
Physical Asset	DMTF	1.0	Optional	See 7.9.
Profile Registration	DMTF	1.0	Mandatory	

277 **6 Description**

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

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

282 The CIM\_Processor class describes the processor characteristics; the CIM\_ProcessorCapabilities class  
 283 describes the capabilities of the processor. The cores of the processor are represented by the  
 284 CIM\_ProcessorCore class and are associated to the CIM\_Processor class through the  
 285 CIM\_ConcreteComponent association. When a core supports hardware threads, the  
 286 CIM\_HardwareThread class is used to represent the thread's properties. The cache memory can be  
 287 associated with either a processor or a core. The cache memory is represented by the CIM\_Memory  
 288 class and is associated to the CIM\_ProcessorCore and CIM\_Processor classes through the  
 289 CIM\_AssociatedCacheMemory class. The physical aspects of a processor are represented through an  
 290 instance of CIM\_Chip class. When the cache memory is off-chip/external, the cache memory's physical  
 291 aspects are represented by an instance of CIM\_PhysicalMemory. Also, the class diagram shows the  
 292 scoping managed system and the profile registration classes.



293

294

**Figure 1 – CPU Profile: Class Diagram**

## 295 7 Implementation

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

### 299 7.1 CIM\_Processor

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

### 301 7.2 Processor Capabilities

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

#### 305 7.2.1 Multi-Core or Multi-Thread Processor Capabilities

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

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

313 **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

#### 314 7.2.2 Single-Core and Single-Thread Processor Capabilities

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

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

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

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

#### 323 7.2.3 CIM\_ProcessorCapabilities.RequestedStatesSupported

324 The RequestedStatesSupported property is an array that contains the supported requested states for the  
 325 instance of CIM\_Processor. This property shall be the super set of the values to be used as the  
 326 RequestedState parameter in the RequestStateChange() method (see 8.1). The value of the

327 CIM\_ProcessorCapabilities.RequestedStatesSupported property shall be an empty array or contain any  
328 combination of the following values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

#### 329 **7.2.4 CIM\_ProcessorCapabilities.ElementNameEditSupported**

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

#### 332 **7.2.5 CIM\_ProcessorCapabilities.MaxElementNameLen**

333 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
334 property has a value of TRUE.

### 335 **7.3 Processor State Management**

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

#### 339 **7.3.1 Processor State Management Support**

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

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

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

### 348 **7.4 CIM\_Processor.RequestedState**

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

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

#### 358 **7.4.1 RequestedState — 12 (Not Applicable) Value**

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

#### 361 **7.4.2 RequestedState — 5 (No Change) Value**

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

## 364 7.5 Modeling the Current Enabled State of the Processor

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

### 368 7.5.1 CIM\_Processor.CPUStatus

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

375 **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	Processor shall be enabled but idling.

### 376 7.5.2 CIM\_Processor.EnabledState

377 The CIM\_Processor.EnabledState property shall be implemented in addition to the  
 378 CIM\_Processor.CPUStatus property. When the CPUStatus property has a value that matches a value in  
 379 the "CPUStatus Value" column in Table 4, the EnabledState property shall have a value that matches a  
 380 value in the "EnabledState Value" column in the same row in the table.

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

CPUStatus 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	2	Enabled

## 382 7.6 Modeling Individual Processor Cores

383 Modeling the individual processor cores is optional functionality. When individual processor cores are  
 384 modeled, the implementation shall instantiate an instance of CIM\_ProcessorCore to represent each

385 processor core. All the requirements in this clause and its subclauses are applicable when an  
386 implementation instantiates the CIM\_ProcessorCore class.

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

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

## 393 **7.6.1 Processor Core Capabilities**

394 The CIM\_EnabledLogicalElementCapabilities class may be used to model the capabilities of processor  
395 cores. When the CIM\_EnabledLogicalElementCapabilities class is instantiated to represent the processor  
396 core capabilities, the instance of CIM\_EnabledLogicalElementCapabilities shall be associated with the  
397 CIM\_ProcessorCore instance through an instance of CIM\_ElementCapabilities and used for advertising  
398 the capabilities of the CIM\_ProcessorCore instance.

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

### 401 **7.6.1.1 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported**

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

### 407 **7.6.1.2 CIM\_EnabledLogicalElementCapabilities.ElementNameEditSupported**

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

### 410 **7.6.1.3 CIM\_EnabledLogicalElementCapabilities.MaxElementNameLen**

411 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
412 property has a value of TRUE.

## 413 **7.6.2 Processor Core State Management**

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

### 417 **7.6.2.1 Processor Core State Management Support**

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

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

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

### 426 7.6.3 CIM\_ProcessorCore.RequestedState

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

431 When processor core state management is supported and the RequestStateChange() method is  
432 successfully executed, the RequestedState property shall be set to the value of the RequestedState  
433 parameter of the RequestStateChange() method. After the RequestStateChange() method has  
434 successfully executed, the RequestedState and EnabledState properties shall have equal values with the  
435 exception of the transitional requested state 11 (Reset). The value of the RequestedState property may  
436 also change as a result of a request for a change to the processor's enabled state by a non-CIM  
437 implementation.

#### 438 7.6.3.1 RequestedState — 12 (Not Applicable) Value

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

#### 441 7.6.3.2 RequestedState — 5 (No Change) Value

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

### 444 7.6.4 Modeling the Current Enabled State of the Processor Core

445 The current enabled state of the processor core is described by the  
446 CIM\_ProcessorCore.CoreEnabledState and CIM\_ProcessorCore.EnabledState properties. Clauses  
447 7.6.4.1 and 7.6.4.2 detail the requirements for implementing these two properties.

#### 448 7.6.4.1 CIM\_ProcessorCore.CoreEnabledState

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

455 **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.
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.



#### 456 7.6.4.2 CIM\_ProcessorCore.EnabledState

457 The CIM\_ProcessorCore.EnabledState property shall be implemented in addition to the  
 458 CIM\_ProcessorCore.CoreEnabledState property. When the CoreEnabledState property value matches a  
 459 value in the “CoreEnabledState Value” column in Table 6, the EnabledState property shall have the value  
 460 that matches the value in the “EnabledState Value” column in the same row in the table.

461 **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

## 462 7.7 Modeling Individual Hardware Threads

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

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

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

### 474 7.7.1 Hardware Thread Capabilities

475 When the CIM\_EnabledLogicalElementCapabilities class is instantiated to represent the hardware thread  
 476 capabilities, the instance of CIM\_EnabledLogicalElementCapabilities shall be associated with the  
 477 CIM\_HardwareThread instance through an instance of CIM\_ElementCapabilities and used for advertising  
 478 the capabilities of the CIM\_HardwareThread instance.

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

#### 481 7.7.1.1 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported

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

**487 7.7.1.2 CIM\_EnabledLogicalElementCapabilities.ElementNameEditSupported**

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

**490 7.7.1.3 CIM\_EnabledLogicalElementCapabilities.MaxElementNameLen**

491 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
492 property has a value of TRUE.

**493 7.7.2 Hardware Thread State Management**

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

**497 7.7.2.1 Hardware Thread State Management Support**

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

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

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

**506 7.7.3 CIM\_HardwareThread.RequestedState**

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

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

**518 7.7.3.1 RequestedState — 12 (Not Applicable) Value**

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

**521 7.7.3.2 RequestedState — 5 (No Change) Value**

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

#### 524 7.7.4 CIM\_HardwareThread.EnabledState

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

531 **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.

### 532 7.8 Modeling Cache Memory

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

536 A single instance of CIM\_Memory shall exist for each discrete cache memory. When the cache memory is  
 537 shared, the single instance of CIM\_Memory shall be associated with multiple instances of CIM\_Processor  
 538 or CIM\_ProcessorCore. When the cache memory is not shared, the instance of CIM\_Memory shall be  
 539 associated with exactly one instance of CIM\_Processor or CIM\_ProcessorCore.

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

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

#### 547 7.8.1 Cache Memory Capabilities

548 When the CIM\_EnabledLogicalElementCapabilities class is instantiated to represent the cache memory  
 549 capabilities, the instance of CIM\_EnabledLogicalElementCapabilities shall be associated with the  
 550 CIM\_Memory instance through an instance of CIM\_ElementCapabilities and used for advertising the  
 551 capabilities of the CIM\_Memory instance.

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

##### 554 7.8.1.1 CIM\_EnabledLogicalElementCapabilities.RequestedStatesSupported

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

**560 7.8.1.2 CIM\_EnabledLogicalElementCapabilities.ElementNameEditSupported**

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

**563 7.8.1.3 CIM\_EnabledLogicalElementCapabilities.MaxElementNameLen**

564 The MaxElementNameLen property shall be implemented when the ElementNameEditSupported  
565 property has a value of TRUE.

**566 7.8.2 Cache Memory State Management**

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

**570 7.8.2.1 Cache Memory State Management Support**

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

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

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

**579 7.8.3 CIM\_Memory.RequestedState**

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

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

**590 7.8.3.1 RequestedState — 12 (Not Applicable) Value**

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

**593 7.8.3.2 RequestedState — 5 (No Change) Value**

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

596 **7.8.4 CIM\_Memory.EnabledState**

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

603 **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.

604 **7.9 Modeling Physical Aspects of Processor and Cache Memory**

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

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

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

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

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

618 **8 Methods**

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

621 **8.1 CIM\_Processor.RequestStateChange()**

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

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

626 When processor state management is supported, the RequestStateChange() method shall be  
 627 implemented and shall not return a value of 1 (Not Supported) (see 7.3.1).

628 Invoking the CIM\_Processor.RequestStateChange() method multiple times could result in earlier  
 629 requests being overwritten or lost.

630 No standard messages are defined for this method.

631 **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

632 **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

## 633 **8.2 CIM\_ProcessorCore.RequestStateChange()**

634 Invocation of the CIM\_ProcessorCore.RequestStateChange() method changes the element's state to the  
635 value that is specified in the RequestedState parameter.

636 Return code values for the RequestStateChange() method shall be as specified in Table 11. Parameters  
637 of the RequestStateChange() method are specified in Table 12.

638 When processor core state management is supported, the RequestStateChange() method shall be  
639 implemented and shall not return a value of 1 (Not Supported) (see 7.6.2.1).

640 Invoking the CIM\_ProcessorCore.RequestStateChange() method multiple times could result in earlier  
641 requests being overwritten or lost.

642 No standard messages are defined for this method.

643 **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

644

**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

645 **8.3 CIM\_HardwareThread.RequestStateChange()**

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

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

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

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

654 No standard messages are defined for this method.

655 **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

656

**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

## 657 8.4 CIM\_Memory.RequestStateChange()

658 Invocation of the CIM\_Memory.RequestStateChange() method changes the element's state to the value  
659 that is specified in the RequestedState parameter.

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

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

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

666 No standard messages are defined for this method.

667 **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

668 **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

## 669 8.5 Profile Conventions for Operations

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

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

673 The default list of operations is as follows:

- 674 • Associators( )
- 675 • AssociatorNames( )
- 676 • EnumerateInstances( )



- 677 • EnumerateInstanceNames( )
- 678 • GetInstance( )
- 679 • References( )
- 680 • ReferenceNames( )

## 681 8.6 CIM\_AssociatedCacheMemory

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

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

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

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

## 687 8.7 CIM\_ConcreteComponent — References CIM\_HardwareThread and 688 CIM\_Processor

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

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

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

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

## 694 8.8 CIM\_ConcreteComponent — References CIM\_ProcessorCore and 695 CIM\_Processor

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

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

700

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

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

701

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

702

703

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](#).

704

705

706

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

707

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

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

708

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

709

710

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](#).

711

712

713

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

714

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

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

715

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

716

717

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](#).

718

719

720

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

721

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

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

722

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

723

724

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](#).

725

726

727

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

728

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

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

729

**8.13 CIM\_EnabledLogicalElementCapabilities**

730

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

731

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

732

**8.14 CIM\_HardwareThread**

733

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](#).

734

735

736

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

737

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

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.14.1.	None

738

**8.14.1 CIM\_HardwareThread — ModifyInstance**

739

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

740

741

The ModifyInstance operation shall be supported and the CIM\_HardwareThread.ElementName property shall be modifiable when the ElementNameEditSupported property of the CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_HardwareThread instance has a value of TRUE. See 7.7.1.2.

742

743

744

## 745 8.15 CIM\_Memory

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

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

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

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.15.1.	None

### 751 8.15.1 CIM\_Memory — ModifyInstance

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

754 The ModifyInstance operation shall be supported and the CIM\_Memory.ElementName property shall be  
 755 modifiable when the ElementNameEditSupported property of the  
 756 CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_Memory instance has  
 757 a value of TRUE. See clause 7.8.1.2.

## 758 8.16 CIM\_Processor

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

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

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

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.16.1.	None

### 764 8.16.1 CIM\_Processor — ModifyInstance

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

767 The ModifyInstance operation shall be supported and the CIM\_Processor.ElementName property shall be  
 768 modifiable when the ElementNameEditSupported property of the  
 769 CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_Processor instance  
 770 has a value of TRUE. See 7.2.4.

## 771 8.17 CIM\_ProcessorCapabilities

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

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

774 **8.18 CIM\_ProcessorCore**

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

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

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

Operation	Requirement	Messages
ModifyInstance	Optional. See 8.18.1.	None

780 **8.18.1 CIM\_ProcessorCore — ModifyInstance**

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

783 The ModifyInstance operation shall be supported and the CIM\_ProcessorCore.ElementName property  
 784 shall be modifiable when the ElementNameEditSupported property of the  
 785 CIM\_EnabledLogicalElementCapabilities instance that is associated with the CIM\_ProcessorCore  
 786 instance has a value of TRUE. See 7.6.1.2.

787 **8.19 CIM\_SystemDevice**

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

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

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

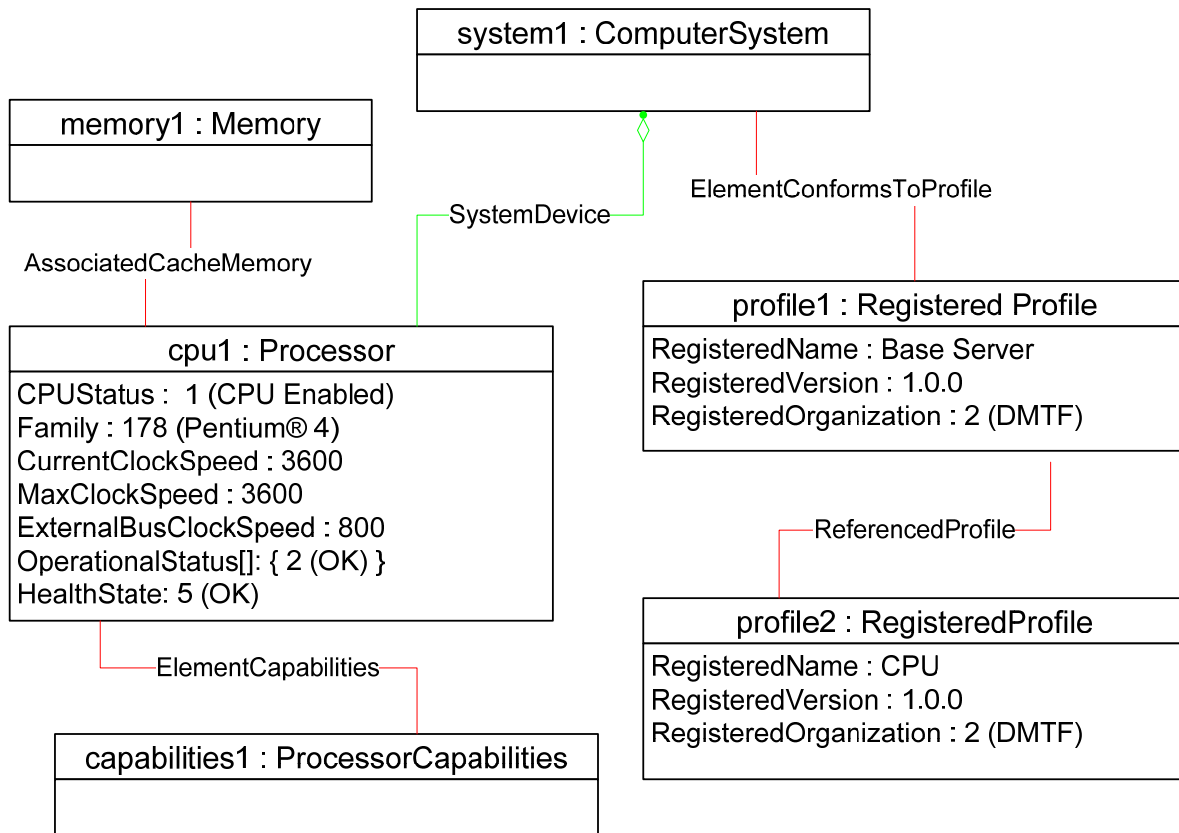
Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

793 **9 Use Cases**

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

795 **9.1 Object Diagrams**

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

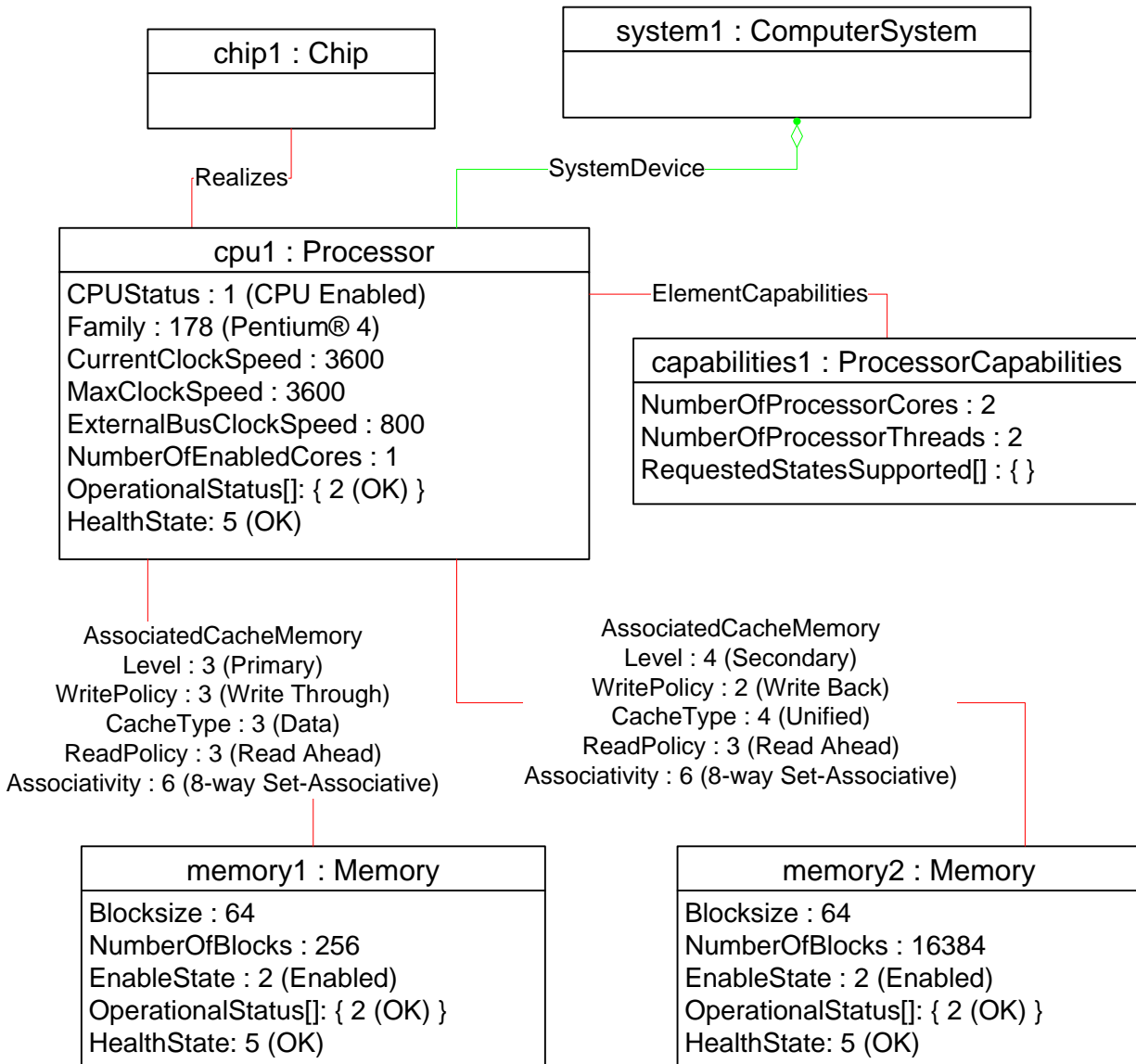


800

801

**Figure 2 – Registered Profile**

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

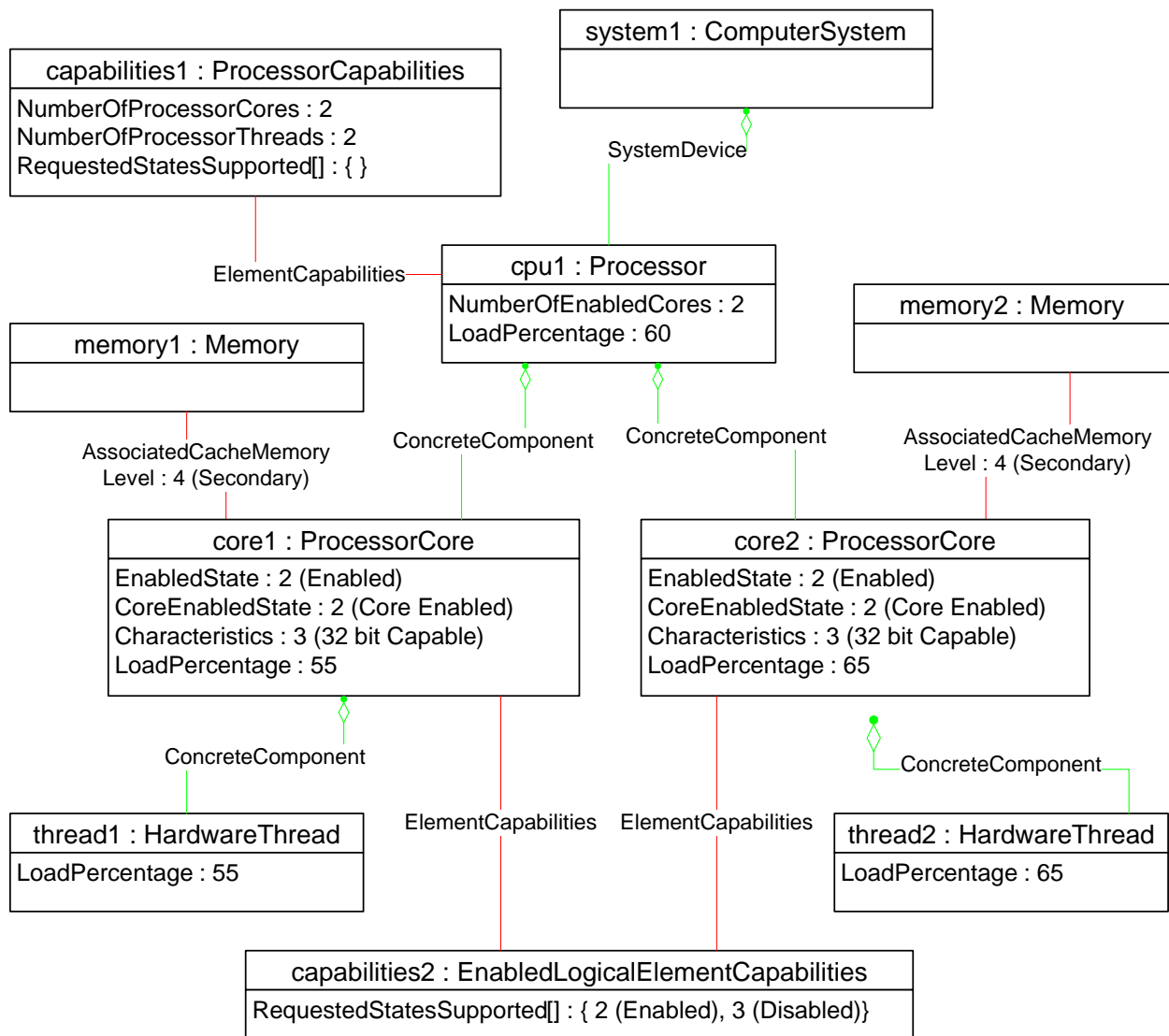


812

813

**Figure 3 – Multi-Core CPU**

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



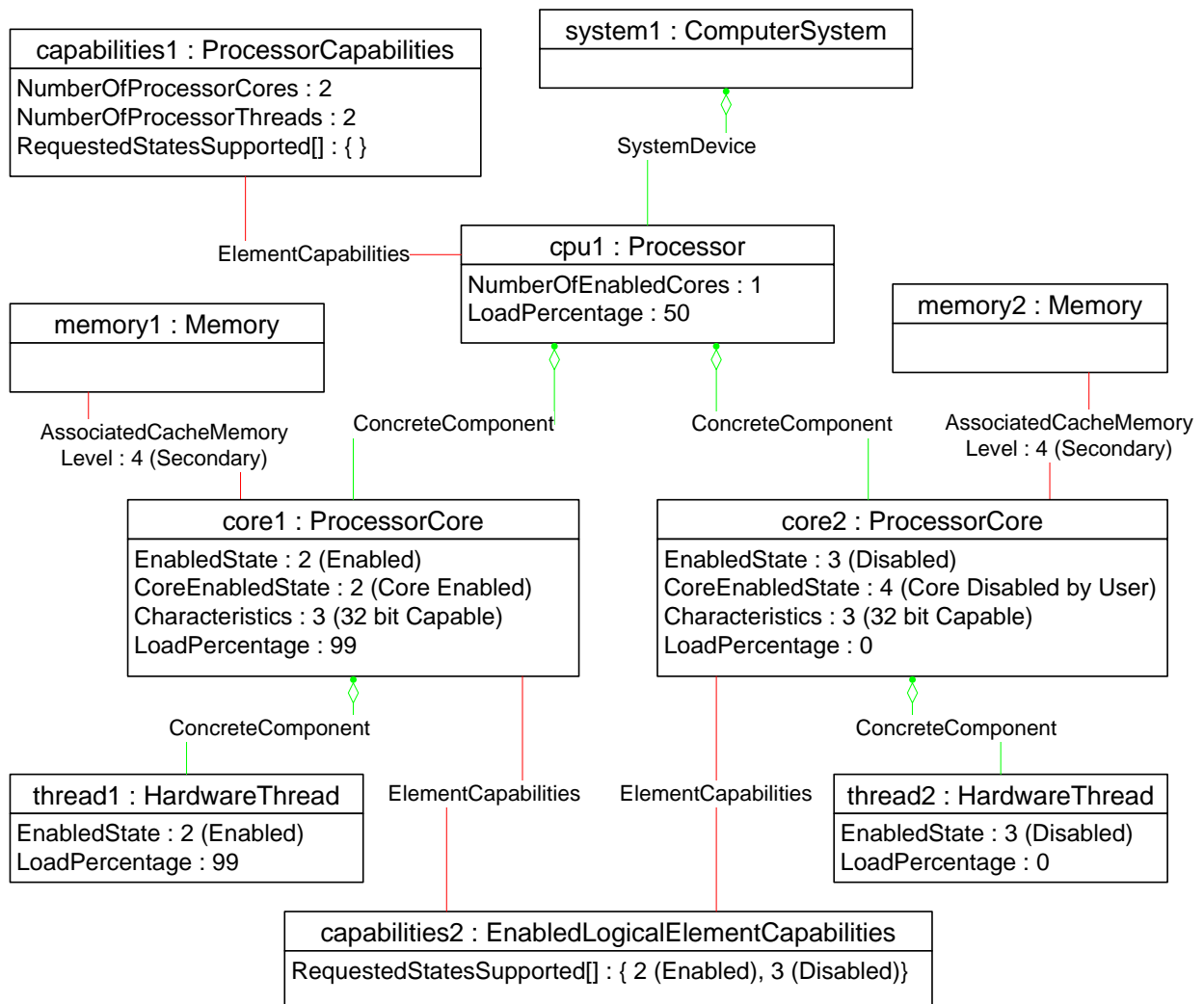
823

824

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



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



829

830

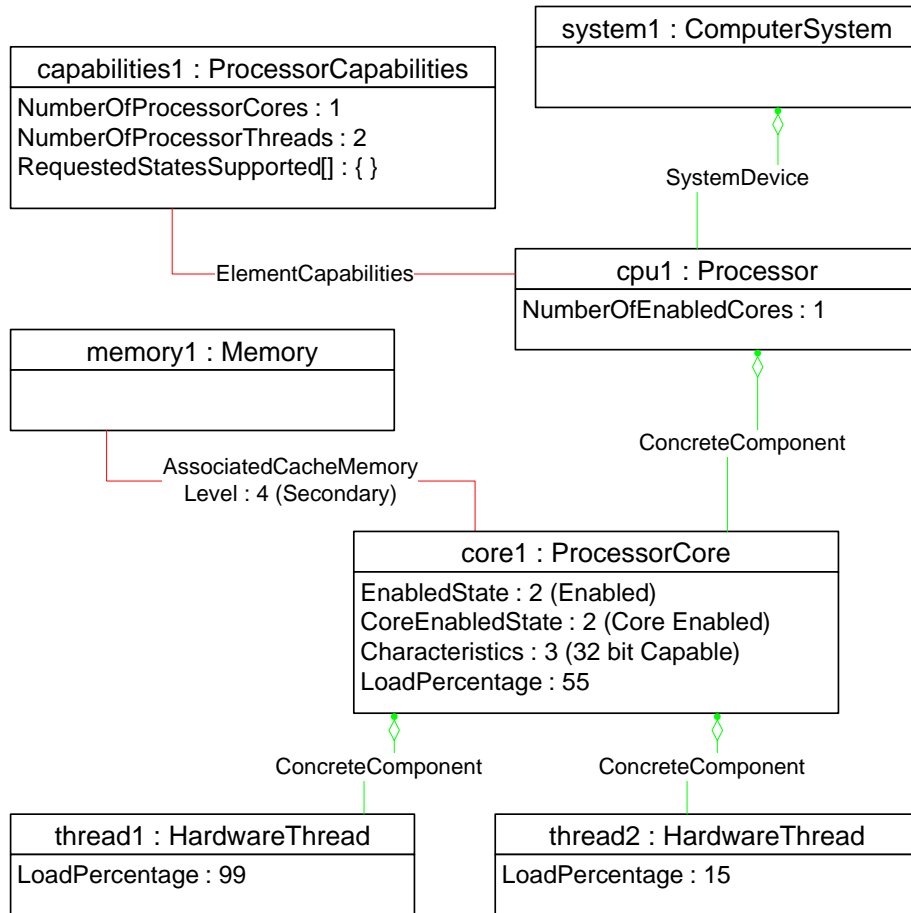
**Figure 5 – Multi-core CPU with a Disabled Core**

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

836 **EXPERIMENTAL**

837 The load percentage calculation is implementation specific; the LoadPercentage property value for core1  
 838 is calculated by taking the average of the values of the LoadPercentage property of thread1 and thread2.

839 **EXPERIMENTAL**

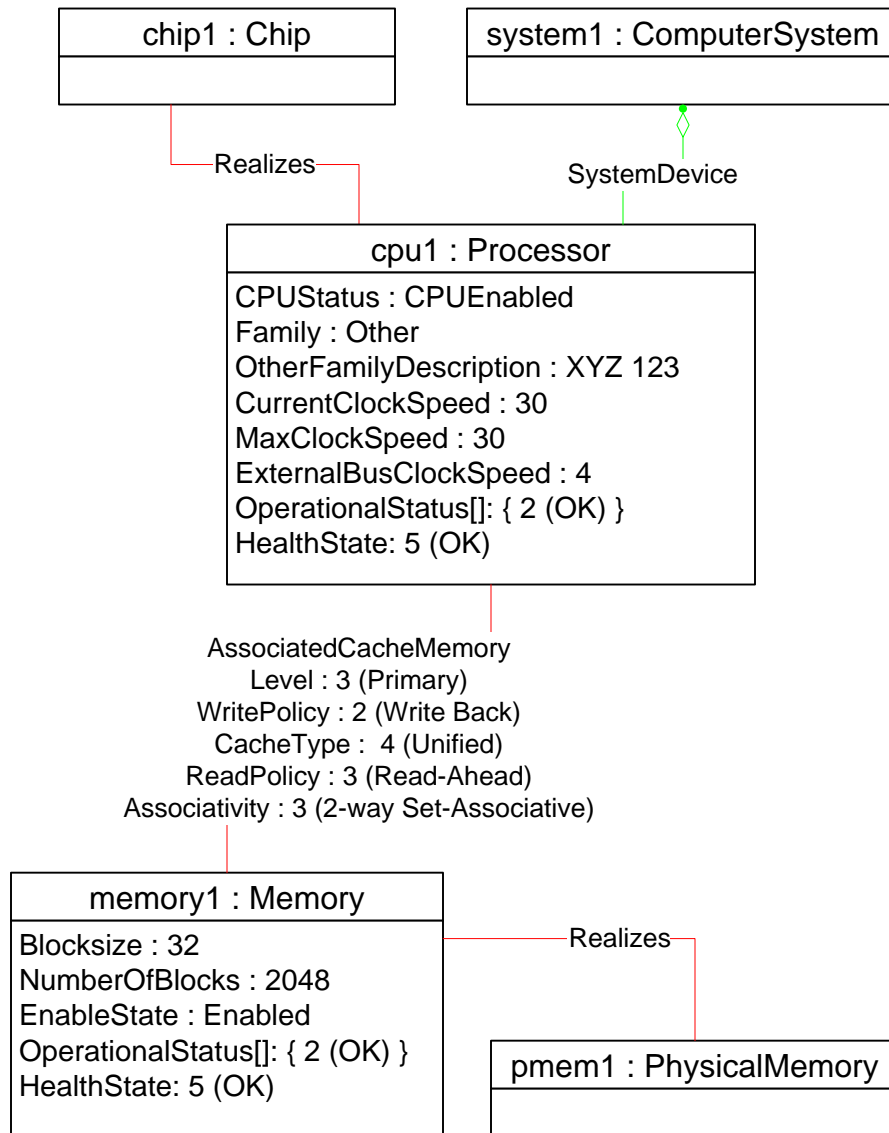


840

841

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

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



846

847

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

848 **9.2 Change the Enabled State of the Memory to the Desired State**

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

- 850 1) Select the instance of CIM\_Memory.  
 851 2) Select the associated instance of CIM\_EnabledLogicalElementCapabilities and verify whether  
 852 the RequestedStatesSupported property contains the desired state.

- 853           3) If the RequestedStatesSupported property contains the desired state, select the instance of  
854           CIM\_Memory and execute the RequestStateChange( ) method with the desired state as a  
855           RequestedState parameter.

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

### 858 **9.3 Change the Enabled State of the CPU to the Desired State**

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

- 860           1) Select the instance of CIM\_Processor.  
861           2) Select the associated instance of CIM\_ProcessorCapabilities and verify whether the  
862           RequestedStatesSupported property contains the desired state.  
863           3) If the RequestedStatesSupported property contains the desired state, select the instance of  
864           CIM\_Processor and execute the RequestStateChange( ) method with the desired state as a  
865           RequestedState parameter.

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

### 868 **9.4 Change the Enabled State of the CPU's Core to the Desired State**

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

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

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

### 878 **9.5 Change the Enabled State of the CPU's Hardware Thread to the Desired 879 State**

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

- 881           1) Select the instance of CIM\_HardwareThread.  
882           2) Select the associated instance of CIM\_EnabledLogicalElementCapabilities and verify whether  
883           the RequestedStatesSupported property contains the desired state.  
884           3) If the RequestedStatesSupported property contains the desired state, select the instance of  
885           CIM\_ProcessorThread and execute the RequestStateChange( ) method with the desired state  
886           as a RequestedState parameter.

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

889 **9.6 Retrieve All the Processor Cores for the CPU**

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

893 **9.7 Retrieve All the Hardware Threads for the CPU**

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

- 895 1) Select all the CIM\_ProcessorCore instances that are associated with the given instance of  
 896 CIM\_Processor through instances of CIM\_Component.
- 897 2) For each instance of CIM\_ProcessorCore, select the instances of CIM\_HardwareThread that  
 898 are associated through instances of CIM\_Component.

899 **9.8 Retrieve CPU’s Cache Memory Information for the CPU**

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

- 901 1) Select all the instances of CIM\_ProcessorCore that are associated with the given instance of  
 902 CIM\_Processor through instances of CIM\_Component.
- 903 2) If no instance of CIM\_ProcessorCore exists, select the instances of  
 904 CIM\_AssociatedCacheMemory that reference the given instance of CIM\_Processor, as well as  
 905 all the instances of CIM\_Memory that are associated with the given instance of CIM\_Processor  
 906 through instances of CIM\_AssociatedCacheMemory.
- 907 3) Otherwise, for each instance of CIM\_ProcessorCore, select the instances of  
 908 CIM\_AssociatedCacheMemory that reference the instance of CIM\_ProcessorCore, as well as  
 909 all the instances of CIM\_Memory that are associated with the instance of CIM\_ProcessorCore  
 910 through instances of CIM\_AssociatedCacheMemory.

911 **10 CIM Elements**

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

915 **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.

Element Name	Requirement	Description
CIM_ElementCapabilities (references CIM_Memory and CIM_EnabledLogicalElementCapabilities)	Optional	See 10.5.
CIM_ElementCapabilities (references CIM_Processor and CIM_ProcessorCapabilities)	Optional	See 10.6.
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		

916 **10.1 CIM\_AssociatedCacheMemory**

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

920 **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).

921 **10.2 CIM\_ConcreteComponent — References CIM\_HardwareThread and**  
 922 **CIM\_ProcessorCore**

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

927 **Table 31 – Class: CIM\_ConcreteComponent — References CIM\_HardwareThread and**  
 928 **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.

929 **10.3 CIM\_ConcreteComponent — References CIM\_ProcessorCore and**  
 930 **CIM\_Processor**

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

935 **Table 32 – Class: CIM\_ConcreteComponent — References CIM\_ProcessorCore and**  
 936 **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.

937 **10.4 CIM\_ElementCapabilities — References CIM\_HardwareThread and**  
 938 **CIM\_EnabledLogicalElementCapabilities**

939 CIM\_ElementCapabilities associates an instance of CIM\_HardwareThread with the instance of  
 940 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of  
 941 CIM\_HardwareThread.

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

945 **Table 33 – Class: CIM\_ElementCapabilities — References CIM\_HardwareThread and**  
 946 **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.

947 **10.5 CIM\_ElementCapabilities — References CIM\_Memory and**  
 948 **CIM\_EnabledLogicalElementCapabilities**

949 CIM\_ElementCapabilities associates an instance of CIM\_Memory with the instance of  
 950 CIM\_EnabledLogicalElementCapabilities that describes the capabilities of the instance of CIM\_Memory.

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



954  
955

**Table 34 – Class: CIM\_ElementCapabilities — References CIM\_Memory and 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.

956  
957

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

958  
959

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

960  
961

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

962  
963

**Table 35 – Class: CIM\_ElementCapabilities — References CIM\_Processor and 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.

964  
965

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

966  
967  
968

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

969  
970  
971

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

972  
973

**Table 36 – Class: CIM\_ElementCapabilities — References CIM\_ProcessorCore and 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.

974 **10.8 CIM\_EnabledLogicalElementCapabilities**

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

977 **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.

978 **10.9 CIM\_HardwareThread**

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

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

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
LoadPercentage	Optional	EXPERIMENTAL
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.

982 **10.10 CIM\_Memory**

983 CIM\_Memory represents the CPU's cache memory. Table 39 contains the requirements for elements of  
984 this class.

985 **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

Elements	Requirement	Notes
ElementName	Mandatory	The property shall match the pattern “.*”.
RequestStateChange()	Conditional	See 8.4.

986 **10.11 CIM\_Processor**

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

989 **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.
LoadPercentage	Optional	None
EnabledState	Mandatory	See 7.5.2.
RequestedState	Mandatory	See 7.4.
OperationalStatus	Mandatory	None
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.

## 990 10.12 CIM\_ProcessorCapabilities

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

993 **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.

## 994 10.13 CIM\_ProcessorCore

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

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

Elements	Requirement	Notes
InstanceID	Mandatory	<b>Key</b>
CoreEnabledState	Mandatory	See 7.6.4.1.
LoadPercentage	Optional	EXPERIMENTAL
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.

## 998 10.14 CIM\_RegisteredProfile

999 The CIM\_RegisteredProfile class is defined by the [Profile Registration Profile](#). The requirements denoted  
1000 in Table 43 are in addition to those mandated by the [Profile Registration Profile](#).

1001 **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 “1.0.1”.
RegisteredOrganization	Mandatory	This property shall have a value of 2 (DMTF).

1002 NOTE: Previous versions of this document included the suffix “Profile” for the RegisteredName value. If  
1003 implementations querying for the RegisteredName value find the suffix “Profile”, they should ignore the suffix, with  
1004 any surrounding white spaces, before any comparison is done with the value as specified in this document.

1005 **10.15 CIM\_SystemDevice**

1006 CIM\_SystemDevice associates an instance of CIM\_Processor with the instance of CIM\_ComputerSystem  
 1007 of which the CIM\_Processor instance is a member. Table 44 contains the requirements for elements of  
 1008 this class.

1009 **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.

1010

1011  
1012  
1013  
1014

## ANNEX A (informative)

### Change Log

Version	Date	Description
1.0.0c	2006-07-02	Preliminary Version of the Profile
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

1015  
1016