



1

2

3

4

Document Number: DSP1022

Date: 2008-10-31

Version: 1.0.0

5 **CPU Profile**

6 **Document Type: Specification**

7 **Document Status: Final Standard**

8 **Document Language: E**

9

10 Copyright Notice

11 Copyright © 2008 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 for uses consistent with this purpose, provided that correct attribution is given. As DMTF
15 specifications may be revised from time to time, the particular version and release date should always be
16 noted.

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

CONTENTS

31	Foreword	6
32	Introduction	7
33	1 Scope	9
34	2 Normative References.....	9
35	2.1 Approved References	9
36	2.2 References under Development	9
37	2.3 Other References.....	9
38	3 Terms and Definitions	9
39	4 Symbols and Abbreviated Terms	11
40	5 Synopsis	11
41	6 Description	12
42	7 Implementation.....	13
43	7.1 CIM_Processor	13
44	7.2 Processor Capabilities	13
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	14
48	7.6 Modeling Individual Processor Cores	15
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	
62	CIM_EnabledLogicalElementCapabilities.....	27
63	8.10 CIM_ElementCapabilities—References CIM_Memory and	
64	CIM_EnabledLogicalElementCapabilities.....	27
65	8.11 CIM_ElementCapabilities—References CIM_Processor and CIM_ProcessorCapabilities.....	27
66	8.12 CIM_ElementCapabilities—References CIM_ProcessorCore and	
67	CIM_EnabledLogicalElementCapabilities.....	28
68	8.13 CIM_EnabledLogicalElementCapabilities.....	28
69	8.14 CIM_HardwareThread	28
70	8.15 CIM_Memory	28
71	8.16 CIM_Processor	29
72	8.17 CIM_ProcessorCapabilities	29
73	8.18 CIM_ProcessorCore	29
74	8.19 CIM_SystemDevice	30
75	9 Use Cases.....	30
76	9.1 Object Diagrams	30
77	9.2 Change the Enabled State of the Memory to the Desired State	35
78	9.3 Change the Enabled State of the CPU to the Desired State	36
79	9.4 Change the Enabled State of the CPU's Core to the Desired State	36
80	9.5 Change the Enabled State of the CPU's Hardware Thread to the Desired State	36
81	9.6 Retrieve All the Processor Cores for the CPU.....	36
82	9.7 Retrieve All the Hardware Threads for the CPU.....	36

83	9.8	Retrieve CPU's Cache Memory Information for the CPU.....	37
84	10	CIM Elements	37
85	10.1	CIM_AssociatedCacheMemory	38
86	10.2	CIM_ConcreteComponent—References CIM_HardwareThread and CIM_ProcessorCore	38
87	10.3	CIM_ConcreteComponent—References CIM_ProcessorCore and CIM_Processor	39
88	10.4	CIM_ElementCapabilities—References CIM_HardwareThread and	
89		CIM_EnabledLogicalElementCapabilities.....	39
90	10.5	CIM_ElementCapabilities—References CIM_Memory and	
91		CIM_EnabledLogicalElementCapabilities.....	39
92	10.6	CIM_ElementCapabilities—References CIM_Processor and CIM_ProcessorCapabilities.....	40
93	10.7	CIM_ElementCapabilities—References CIM_ProcessorCore and	
94		CIM_EnabledLogicalElementCapabilities.....	40
95	10.8	CIM_EnabledLogicalElementCapabilities.....	41
96	10.9	CIM_HardwareThread	41
97	10.10	CIM_Memory	41
98	10.11	CIM_Processor	42
99	10.12	CIM_ProcessorCapabilities	42
100	10.13	CIM_ProcessorCore	43
101	10.14	CIM_RegisteredProfile.....	43
102	10.15	CIM_SystemDevice	43
103		ANNEX A (informative) Change Log.....	44
104		ANNEX B (informative) Acknowledgments.....	45
105			

106 Figures

107	Figure 1 – CPU Profile: Class Diagram	12
108	Figure 2 – Registered Profile	30
109	Figure 3 – Multi-core CPU.....	31
110	Figure 4 – Detailed Multi-core CPU	32
111	Figure 5 – Multi-core CPU with a Disabled Core	33
112	Figure 6 – Single Core, Multi-hardware Thread CPU	34
113	Figure 7 – Processor with Off-Chip Cache	35
114		

115 Tables

116	Table 1 – Related Profiles.....	11
117	Table 2 – CIM_ProcessorCapabilities Properties Mapping to SMBIOS Equivalence	13
118	Table 3 – CIM_Processor.CPUStatus Value Descriptions	15
119	Table 4 – Mapping for CPUStatus Property and EnabledState Property Values	15
120	Table 5 – CIM_ProcessorCore.CoreEnabledState Value Descriptions.....	17
121	Table 6 – Mapping for the CoreEnabledState Property and EnabledState Property Values	18
122	Table 7 – CIM_HardwareThread.EnabledState Value Descriptions	20
123	Table 8 – CIM_Memory.EnabledState Value Descriptions.....	22
124	Table 9 – CIM_Processor.RequestStateChange() Method: Return Code Values.....	23
125	Table 10 – CIM_Processor.RequestStateChange() Method: Parameters	23
126	Table 11 – CIM_ProcessorCore.RequestStateChange() Method: Return Code Values.....	23
127	Table 12 – CIM_ProcessorCore.RequestStateChange() Method: Parameters.....	24
128	Table 13 – CIM_HardwareThread.RequestStateChange() Method: Return Code Values.....	24
129	Table 14 – CIM_HardwareThread.RequestStateChange() Method: Parameters.....	24

130 Table 15 – CIM_Memory.RequestStateChange() Method: Return Code Values..... 25

131 Table 16 – CIM_Memory.RequestStateChange() Method: Parameters..... 25

132 Table 17 – Operations: CIM_AssociatedCacheMemory..... 26

133 Table 18 – Operations: CIM_ConcreteComponent 26

134 Table 19 – Operations: CIM_ConcreteComponent 26

135 Table 20 – Operations: CIM_ElementCapabilities 27

136 Table 21 – Operations: CIM_ElementCapabilities 27

137 Table 22 – Operations: CIM_ElementCapabilities 27

138 Table 23 – Operations: CIM_ElementCapabilities 28

139 Table 24 – Operations: CIM_HardwareThread..... 28

140 Table 25 – Operations: CIM_Memory..... 28

141 Table 26 – Operations: CIM_Processor..... 29

142 Table 27 – Operations: CIM_ProcessorCore..... 29

143 Table 28 – Operations: CIM_SystemDevice..... 30

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

145 Table 30 – Class: CIM_AssociatedCacheMemory 38

146 Table 31 – Class: CIM_ConcreteComponent—References CIM_HardwareThread and
147 CIM_ProcessorCore 39

148 Table 32 – Class: CIM_ConcreteComponent—References CIM_ProcessorCore and CIM_Processor 39

149 Table 33 – Class: CIM_ElementCapabilities—References CIM_HardwareThread and
150 CIM_EnabledLogicalElementCapabilities..... 39

151 Table 34 – Class: CIM_ElementCapabilities—References CIM_Memory and
152 CIM_EnabledLogicalElementCapabilities..... 40

153 Table 35 – Class: CIM_ElementCapabilities—References CIM_Processor and
154 CIM_ProcessorCapabilities 40

155 Table 36 – Class: CIM_ElementCapabilities—References CIM_ProcessorCore and
156 CIM_EnabledLogicalElementCapabilities..... 40

157 Table 37 – Class: CIM_EnabledLogicalElementCapabilities..... 41

158 Table 38 – Class: CIM_HardwareThread 41

159 Table 39 – Class: CIM_Memory 41

160 Table 40 – Class: CIM_Processor 42

161 Table 41 – Class: CIM_ProcessorCapabilities..... 42

162 Table 42 – Class: CIM_ProcessorCore 43

163 Table 43 – Class: CIM_RegisteredProfile..... 43

164 Table 44 – Class: CIM_SystemDevice 43

165

166

Foreword

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

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

170

Introduction

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

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

177

CPU Profile

178 1 Scope

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

182 2 Normative References

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

186 2.1 Approved References

187 DMTF [DSP0004](#), *CIM Infrastructure Specification 2.3.0*

188 DMTF [DSP1033](#), *Profile Registration Profile 1.0*

189 DMTF [DSP0200](#), *CIM Operations over HTTP 1.2.0*

190 DMTF [DSP1000](#), *Management Profile Specification Template*

191 DMTF [DSP1001](#), *Management Profile Specification Usage Guide*

192 2.2 References under Development

193 DMTF [DSP0134](#), *System Management BIOS (SMBIOS) Reference Specification 2.6*

194 2.3 Other References

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

196 OMG, [Unified Modeling Language \(UML\) from the Open Management Group \(OMG\)](#)

197 IETF [rfc2234](#), *Augmented BNF for Syntax Specifications: ABNF*

198 3 Terms and Definitions

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

201 3.1

202 **can**

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

204 3.2

205 **cannot**

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

- 207 **3.3**
208 **conditional**
209 indicates requirements to be followed strictly to conform to the document when the specified conditions
210 are met
- 211 **3.4**
212 **mandatory**
213 indicates requirements to be followed strictly to conform to the document and from which no deviation is
214 permitted
- 215 **3.5**
216 **may**
217 indicates a course of action permissible within the limits of the document
- 218 **3.6**
219 **need not**
220 indicates a course of action permissible within the limits of the document
- 221 **3.7**
222 **optional**
223 indicates a course of action permissible within the limits of the document
- 224 **3.8**
225 **referencing profile**
226 indicates a profile that owns the definition of this class and can include a reference to this profile in its
227 "Referenced Profiles" table
- 228 **3.9**
229 **shall**
230 indicates requirements to be followed strictly to conform to the document and from which no deviation is
231 permitted
- 232 **3.10**
233 **shall not**
234 indicates requirements to be followed strictly to conform to the document and from which no deviation is
235 permitted
- 236 **3.11**
237 **should**
238 indicates that among several possibilities, one is recommended as particularly suitable, without
239 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 240 **3.12**
241 **should not**
242 indicates that a certain possibility or course of action is deprecated but not prohibited
- 243 **3.13**
244 **unspecified**
245 indicates that this profile does not define any constraints for the referenced CIM element or operation
- 246 **3.14**
247 **Cache Memory**
248 indicates the instance of CIM_Memory that represents the cache memory for the processor

249 **3.15**
 250 **Host Processor**
 251 indicates the instance of CIM_Processor that represents the processor that hosts the processor core

252 **3.16**
 253 **Threading Processor Core**
 254 indicates the instance of CIM_ProcessorCore that represents the processor core that enables the
 255 hardware threading

256 **4 Symbols and Abbreviated Terms**

257 **Experimental Maturity Level**

258 Some of the content considered for inclusion in the *CPU Profile* has yet to receive sufficient review to
 259 satisfy the adoption requirements set forth by the Technical Committee within the DMTF. This content is
 260 presented here as an aid to implementers who are interested in likely future developments within this
 261 specification. The content marked experimental may change as implementation experience is gained.
 262 There is a high likelihood that it will be included in an upcoming revision of the specification. Until that
 263 time, it is purely informational, and is clearly marked within the text.

264 A sample of the typographical convention for experimental content is included here:

265 **EXPERIMENTAL**

266 Experimental content appears here.

267 **EXPERIMENTAL**

268 **5 Synopsis**

269 **Profile Name:** *CPU*

270 **Version:** 1.0.0

271 **Organization:** DMTF

272 **CIM Schema Version:** 2.19.0

273 **Central Class:** CIM_Processor

274 **Scoping Class:** CIM_ComputerSystem

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

277 **Table 1 – Related Profiles**

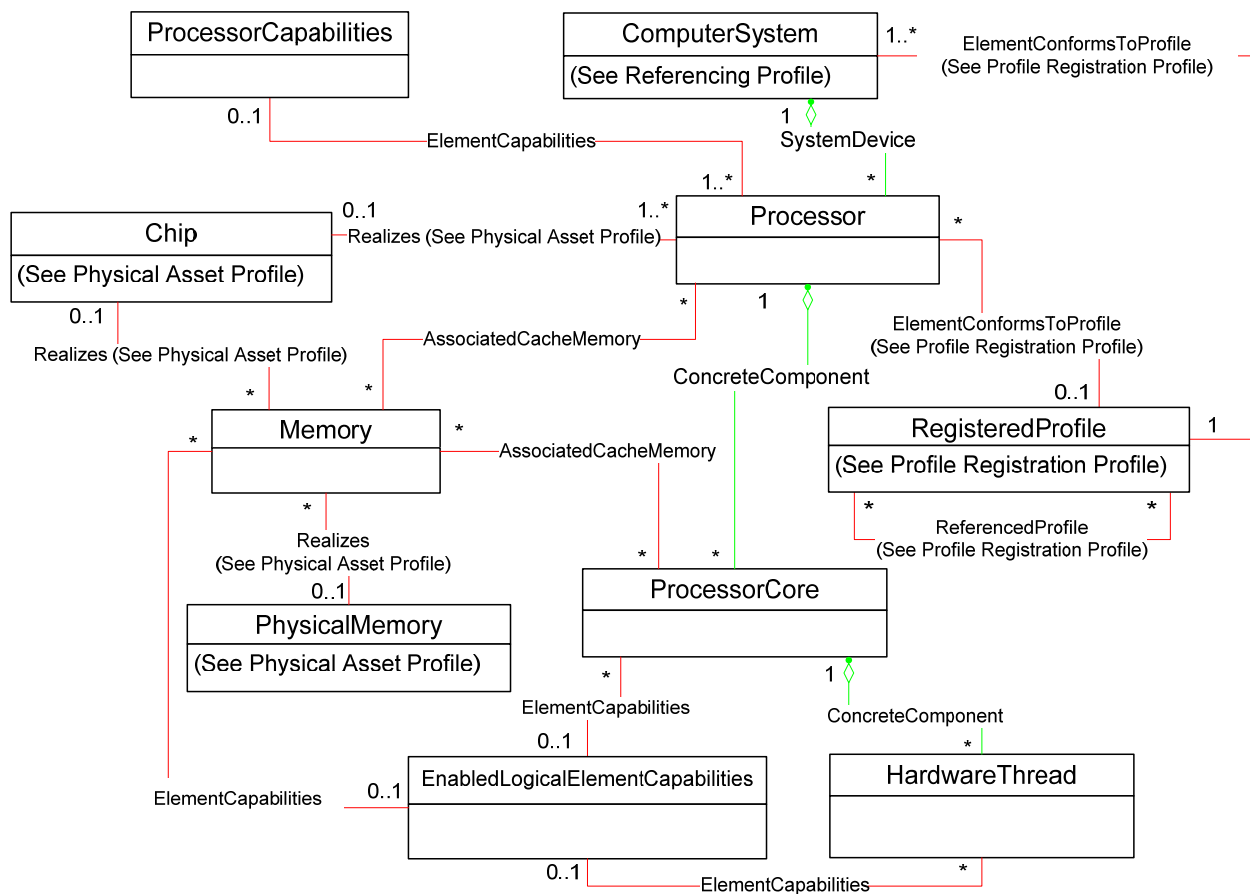
Profile Name	Organization	Version	Requirement	Description
<i>Physical Asset</i>	DMTF	1.0.0	Optional	See section 7.9.
<i>Profile Registration</i>	DMTF	1.0.0	Mandatory	

278 **6 Description**

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

281 Figure 1 represents the class schema for the *CPU Profile*. For simplicity, the prefix CIM_ has been
 282 removed from the names of the classes.

283 The CIM_Processor class describes the processor characteristics; the CIM_ProcessorCapabilities class
 284 describes the capabilities of the processor. The cores of the processor are represented by the
 285 CIM_ProcessorCore class and are associated to the CIM_Processor class through the
 286 CIM_ConcreteComponent association. When a core supports hardware threads, the
 287 CIM_HardwareThread class is used to represent the thread's properties. The cache memory can be
 288 associated with either a processor or a core. The cache memory is represented by the CIM_Memory
 289 class and is associated to the CIM_ProcessorCore and CIM_Processor classes through the
 290 CIM_AssociatedCacheMemory class. The physical aspects of a processor are represented through an
 291 instance of CIM_Chip class. When the cache memory is off-chip/external, the cache memory's physical
 292 aspects are represented by an instance of CIM_PhysicalMemory. Also, the class diagram shows the
 293 scoping managed system and the profile registration classes.



294

295

Figure 1 – CPU Profile: Class Diagram

296 **7 Implementation**

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

300 **7.1 CIM_Processor**

301 Zero or more instances of CIM_Processor shall be instantiated.

302 **7.2 Processor Capabilities**

303 The CIM_ProcessorCapabilities class may be instantiated to represent the processor capabilities. Only
 304 one instance of CIM_ProcessorCapabilities shall be associated with a given instance of CIM_Processor
 305 through an instance of CIM_ElementCapabilities.

306 **7.2.1 Multi-core or Multi-thread Processor Capabilities**

307 When modeling the capabilities of a multi-core or multi-thread processor, the CIM_ProcessorCapabilities
 308 class shall be instantiated and associated to the instance of CIM_Processor that represents the multi-core
 309 or multi-thread processor.

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

314 **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

315 **7.2.2 Single-Core and Single-Thread Processor Capabilities**

316 When modeling the capabilities of a single-core and single-thread processor, the
 317 CIM_ProcessorCapabilities may not be instantiated.

318 When no instance of CIM_ProcessorCapabilities is associated with the instance of CIM_Processor that
 319 represents the processor, the processor is a single-core and single-thread processor.

320 When an instance of CIM_ProcessorCapabilities is associated with the instance of CIM_Processor that
 321 represents the single-core and single-thread processor, the following requirements apply:

- 322 • The CIM_ProcessorCapabilities.NumberOfProcessorCores property shall have a value of 1.
- 323 • The CIM_ProcessorCapabilities.NumberOfHardwareThreads property shall have a value of 1.

324 **7.2.3 CIM_ProcessorCapabilities.RequestedStatesSupported**

325 The RequestedStatesSupported property is an array that contains the supported requested states for the
 326 instance of CIM_Processor. This property shall be the super set of the values to be used as the
 327 RequestedState parameter in the RequestStateChange() method (see section 8.1). The value of the
 328 CIM_ProcessorCapabilities.RequestedStatesSupported property shall be an empty array or contain any
 329 combination of the following values: 2 (Enabled), 3 (Disabled), or 11 (Reset).

330 **7.2.4 CIM_ProcessorCapabilities.ElementNameEditSupported**

331 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
332 client modification of the CIM_Processor.ElementName property.

333 **7.2.5 CIM_ProcessorCapabilities.MaxElementNameLen**

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

336 **7.3 Processor State Management**

337 Processor state management requires that the CIM_Processor.RequestStateChange() method be
338 supported (see section 8.1) and the value of the CIM_Processor.RequestedState property not match 12
339 (Not Applicable).

340 **7.3.1 Processor State Management Support**

341 When the instance of CIM_ProcessorCapabilities does not exist, processor state management shall not
342 be supported.

343 When the value of the CIM_ProcessorCapabilities.RequestedStatesSupported property of the associated
344 CIM_ProcessorCapabilities instance is an empty array, processor state management shall not be
345 supported.

346 When the value of the CIM_ProcessorCapabilities.RequestedStatesSupported property of the associated
347 CIM_ProcessorCapabilities instance is not an empty array, processor state management shall be
348 supported.

349 **7.4 CIM_Processor.RequestedState**

350 The CIM_Processor.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No Change),
351 or a value contained in the CIM_ProcessorCapabilities.RequestedStatesSupported property array of the
352 associated CIM_ProcessorCapabilities instance (see section 7.2.2).

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

359 **7.4.1 RequestedState—12 (Not Applicable) Value**

360 When processor state management is not supported, the value of the CIM_Processor.RequestedState
361 property shall be 12 (Not Applicable).

362 **7.4.2 RequestedState—5 (No Change) Value**

363 When processor state management is supported, the initial value of the CIM_Processor.RequestedState
364 property shall be 5 (No Change).

365 **7.5 Modeling the Current Enabled State of the Processor**

366 The current enabled state of the processor is described by the CIM_Processor.CPUStatus and
367 CIM_Processor.EnabledState properties. Sections 7.5.1 and 7.5.2 detail the requirements for
368 implementing these two properties.

369 **7.5.1 CIM_Processor.CPUStatus**

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

376 **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.

377 **7.5.2 CIM_Processor.EnabledState**

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

382 **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

383 **7.6 Modeling Individual Processor Cores**

384 Modeling the individual processor cores is optional functionality. When individual processor cores are
 385 modeled, the implementation shall instantiate an instance of CIM_ProcessorCore to represent each
 386 processor core. All the requirements in this section and its subsections are applicable when an
 387 implementation instantiates the CIM_ProcessorCore class.

388 Each instance of CIM_ProcessorCore shall be associated through an instance of
 389 CIM_ConcreteComponent to only one instance of CIM_Processor that represents the processor (Host
 390 Processor) that hosts the processor core.

391 The number of instances of CIM_ProcessorCore associated with the Host Processor shall be equal to the
392 value of the CIM_ProcessorCapabilities.NumberOfProcessorCores property of the instance of
393 CIM_ProcessorCapabilities that is associated with the Host Processor.

394 **7.6.1 Processor Core Capabilities**

395 The CIM_EnabledLogicalElementCapabilities class may be used to model the capabilities of processor
396 cores. When the CIM_EnabledLogicalElementCapabilities class is instantiated to represent the processor
397 core capabilities, the instance of CIM_EnabledLogicalElementCapabilities shall be associated with the
398 CIM_ProcessorCore instance through an instance of CIM_ElementCapabilities and used for advertising
399 the capabilities of the CIM_ProcessorCore instance.

400 There shall be at most one instance of CIM_EnabledLogicalElementCapabilities associated with a given
401 instance of CIM_ProcessorCore.

402 **7.6.1.1 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported**

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

408 **7.6.1.2 CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported**

409 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
410 client modification of the CIM_ProcessorCore.ElementName property.

411 **7.6.1.3 CIM_EnabledLogicalElementCapabilities.MaxElementNameLen**

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

414 **7.6.2 Processor Core State Management**

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

418 **7.6.2.1 Processor Core State Management Support**

419 When no CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_ProcessorCore
420 instance, processor core state management shall not be supported.

421 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_ProcessorCore
422 instance but the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
423 property is an empty array, processor core state management shall not be supported.

424 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_ProcessorCore
425 instance and the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
426 property is not an empty array, processor core state management shall be supported.

427 **7.6.3 CIM_ProcessorCore.RequestedState**

428 The CIM_ProcessorCore.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No
429 Change), or a value contained in the
430 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated
431 CIM_EnabledLogicalElementCapabilities instance (see section 7.6.1.1).

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

439 **7.6.3.1 RequestedState—12 (Not Applicable) Value**

440 When processor core state management is not supported, the value of the
 441 CIM_ProcessorCore.RequestedState property shall be 12 (Not Applicable).

442 **7.6.3.2 RequestedState—5 (No Change) Value**

443 When processor core state management is supported, the initial value of the
 444 CIM_ProcessorCore.RequestedState property shall be 5 (No Change).

445 **7.6.4 Modeling the Current Enabled State of the Processor Core**

446 The current enabled state of the processor core is described by the
 447 CIM_ProcessorCore.CoreEnabledState and CIM_ProcessorCore.EnabledState properties. Sections
 448 7.6.4.1 and 7.6.4.2 detail the requirements for implementing these two properties.

449 **7.6.4.1 CIM_ProcessorCore.CoreEnabledState**

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

456 **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.

457 **7.6.4.2 CIM_ProcessorCore.EnabledState**

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

462 **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

463 **7.7 Modeling Individual Hardware Threads**

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

468 The instance of CIM_HardwareThread shall be associated through an instance of
 469 CIM_ConcreteComponent to only one instance of CIM_ProcessorCore that represents the processor core
 470 that enables the hardware thread (Threading Processor Core).

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

475 **7.7.1 Hardware Thread Capabilities**

476 When the CIM_EnabledLogicalElementCapabilities class is instantiated to represent the hardware thread
 477 capabilities, the instance of CIM_EnabledLogicalElementCapabilities shall be associated with the
 478 CIM_HardwareThread instance through an instance of CIM_ElementCapabilities and used for advertising
 479 the capabilities of the CIM_HardwareThread instance.

480 At most one instance of CIM_EnabledLogicalElementCapabilities shall be associated with a given
 481 instance of CIM_HardwareThread.

482 **7.7.1.1 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported**

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

488 7.7.1.2 CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported

489 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
490 client modification of the CIM_HardwareThread.ElementName property.

491 7.7.1.3 CIM_EnabledLogicalElementCapabilities.MaxElementNameLen

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

494 7.7.2 Hardware Thread State Management

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

498 7.7.2.1 Hardware Thread State Management Support

499 When no CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_HardwareThread
500 instance, hardware thread state management shall not be supported.

501 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_HardwareThread
502 instance but the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
503 property is an empty array, hardware thread state management shall not be supported.

504 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_HardwareThread
505 instance and the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported
506 property is not an empty array, hardware thread state management shall be supported.

507 7.7.3 CIM_HardwareThread.RequestedState

508 The CIM_HardwareThread.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No
509 Change), or a value contained in the
510 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property array of the associated
511 CIM_EnabledLogicalElementCapabilities instance (see section 7.7.1.1).

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

519 7.7.3.1 RequestedState—12 (Not Applicable) Value

520 When hardware thread state management is not supported, the value of the
521 CIM_HardwareThread.RequestedState property shall be 12 (Not Applicable).

522 7.7.3.2 RequestedState—5 (No Change) Value

523 When hardware thread state management is supported, the initial value of the
524 CIM_HardwareThread.RequestedState property shall be 5 (No Change).

525 7.7.4 CIM_HardwareThread.EnabledState

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

532 **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.

533 7.8 Modeling Cache Memory

534 Modeling the cache memory of the processor is optional. The implementation may instantiate instances of
 535 CIM_Memory to represent the cache memory. All the requirements in this section and its subsections are
 536 applicable when an implementation instantiates the CIM_Memory class that represents cache memory.

537 A single instance of CIM_Memory shall exist for each discrete cache memory. When the cache memory is
 538 shared, the single instance of CIM_Memory shall be associated with multiple instances of CIM_Processor
 539 or CIM_ProcessorCore. When the cache memory is not shared, the instance of CIM_Memory shall be
 540 associated with exactly one instance of CIM_Processor or CIM_ProcessorCore.

541 When the optional behavior described in section 7.6 is implemented, each instance of CIM_Memory that
 542 represents the cache memory used by the processor core shall be associated with the instance of
 543 CIM_ProcessorCore that represents the processor core through an instance of
 544 CIM_AssociatedCacheMemory and shall not be associated with the Host Processor of the core.

545 When the optional behavior described in section 7.6 is not implemented, each instance of CIM_Memory
 546 that represents the cache memory used by the processor shall be associated through an instance of the
 547 CIM_AssociatedCacheMemory to the instance of CIM_Processor.

548 7.8.1 Cache Memory Capabilities

549 When the CIM_EnabledLogicalElementCapabilities class is instantiated to represent the cache memory
 550 capabilities, the instance of CIM_EnabledLogicalElementCapabilities shall be associated with the
 551 CIM_Memory instance through an instance of CIM_ElementCapabilities and used for advertising the
 552 capabilities of the CIM_Memory instance.

553 At most one instance of CIM_EnabledLogicalElementCapabilities shall be associated with a given
 554 instance of CIM_Memory.

555 7.8.1.1 CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported

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

561 7.8.1.2 CIM_EnabledLogicalElementCapabilities.ElementNameEditSupported

562 The ElementNameEditSupported property shall have a value of TRUE when the implementation supports
563 client modification of the CIM_Memory.ElementName property.

564 7.8.1.3 CIM_EnabledLogicalElementCapabilities.MaxElementNameLen

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

567 7.8.2 Cache Memory State Management

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

571 7.8.2.1 Cache Memory State Management Support

572 When no CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_Memory instance,
573 cache memory state management shall not be supported.

574 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_Memory instance
575 but the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property is an
576 empty array, cache memory state management shall not be supported.

577 When a CIM_EnabledLogicalElementCapabilities instance is associated with the CIM_Memory instance
578 and the value of the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property is not
579 an empty array, cache memory state management shall be supported.

580 7.8.3 CIM_Memory.RequestedState

581 The CIM_Memory.RequestedState property shall have a value of 12 (Not Applicable) or 5 (No Change),
582 or a value contained in the CIM_EnabledLogicalElementCapabilities.RequestedStatesSupported property
583 array of the associated CIM_EnabledLogicalElementCapabilities instance (see section 7.8.1.1).

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

591 7.8.3.1 RequestedState—12 (Not Applicable) Value

592 When cache memory state management is not supported, the value of the CIM_Memory.RequestedState
593 property shall be 12 (Not Applicable).

594 7.8.3.2 RequestedState—5 (No Change) Value

595 When cache memory state management is supported, the initial value of the
596 CIM_Memory.RequestedState property shall be 5 (No Change).

597 7.8.4 CIM_Memory.EnabledState

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

604 **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.

605 7.9 Modeling Physical Aspects of Processor and Cache Memory

606 The *Physical Asset Profile* may be implemented to model the physical aspects of a processor, including
 607 the asset information of the processor or the internal or off-chip cache memory.

608 When the processor's or internal cache memory's physical aspects are represented, a CIM_Chip instance
 609 shall be instantiated and associated with the instance of CIM_Processor or with any instances of
 610 CIM_Memory that represent the internal cache through instances of CIM_Realizes.

611 When the off-chip cache memory is represented along with its physical aspects, a CIM_PhysicalMemory
 612 instance shall be instantiated and associated with the instance of CIM_Memory through an instance of
 613 CIM_Realizes.

614 Note that when processor cores or hardware threads for the processor are modeled with the physical
 615 aspects of the processor, the instances of CIM_ProcessorCore and CIM_HardwareThread shall not be
 616 associated with the instance of CIM_Chip that represents the physical aspects of the processor.

617 The configuration capacity of the managed system for processors may be modeled using the optional
 618 behavior specified in the "Modeling Configuration Capacity" section of the *Physical Asset Profile*.

619 8 Methods

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

622 8.1 CIM_Processor.RequestStateChange()

623 Invocation of the CIM_Processor.RequestStateChange() method changes the element's state to the
 624 value that is specified in the RequestedState parameter.

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

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

629 Invoking the CIM_Processor.RequestStateChange() method multiple times could result in earlier
 630 requests being overwritten or lost.

631 No standard messages are defined for this method.

632 **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

633 **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

634 **8.2 CIM_ProcessorCore.RequestStateChange()**

635 Invocation of the CIM_ProcessorCore.RequestStateChange() method changes the element’s state to the
636 value that is specified in the RequestedState parameter.

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

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

641 Invoking the CIM_ProcessorCore.RequestStateChange() method multiple times could result in earlier
642 requests being overwritten or lost.

643 No standard messages are defined for this method.

644 **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

645 **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

646 **8.3 CIM_HardwareThread.RequestStateChange()**

647 Invocation of the CIM_HardwareThread.RequestStateChange() method changes the element's state to
648 the value that is specified in the RequestedState parameter.

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

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

653 Invoking the CIM_HardwareThread.RequestStateChange() method multiple times could result in earlier
654 requests being overwritten or lost.

655 No standard messages are defined for this method.

656 **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

657 **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

658 **8.4 CIM_Memory.RequestStateChange()**

659 Invocation of the CIM_Memory.RequestStateChange() method changes the element’s state to the value
 660 that is specified in the RequestedState parameter.

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

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

665 Invoking the CIM_Memory.RequestStateChange() method multiple times could result in earlier requests
 666 being overwritten or lost.

667 No standard messages are defined for this method.

668 **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

669 **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

670 **8.5 Profile Conventions for Operations**

671 Support for operations for each profile class (including associations) is specified in the following
 672 subclauses. Each subclause includes either the statement “All operations in the default list in section 8.5
 673 are supported as described by [DSP0200 version 1.2](#)” or a table listing all of the operations that are not
 674 supported by this profile or where the profile requires behavior other than that described by [DSP0200](#)
 675 [version 1.2](#).

676 The default list of operations is as follows:

- 677 • GetInstance
- 678 • EnumerateInstances
- 679 • EnumerateInstanceNames
- 680 • Associators

- 681 • AssociatorNames
- 682 • References
- 683 • ReferenceNames

684 A compliant implementation shall support all the operations in the default list for each class, unless the
685 “Requirement” column states something other than *Mandatory*.

686 8.6 CIM_AssociatedCacheMemory

687 Table 17 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
688 or shall not be supported.

689 **Table 17 – Operations: CIM_AssociatedCacheMemory**

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

690 8.7 CIM_ConcreteComponent—References CIM_HardwareThread and 691 CIM_Processor

692 Table 18 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
693 or shall not be supported.

694 **Table 18 – Operations: CIM_ConcreteComponent**

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

695 8.8 CIM_ConcreteComponent—References CIM_ProcessorCore and 696 CIM_Processor

697 Table 19 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
698 or shall not be supported.

699 **Table 19 – Operations: CIM_ConcreteComponent**

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

700 **8.9 CIM_ElementCapabilities—References CIM_HardwareThread and**
 701 **CIM_EnabledLogicalElementCapabilities**

702 Table 20 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
 703 or shall not be supported.

704 **Table 20 – Operations: CIM_ElementCapabilities**

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

705 **8.10 CIM_ElementCapabilities—References CIM_Memory and**
 706 **CIM_EnabledLogicalElementCapabilities**

707 Table 21 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
 708 or shall not be supported.

709 **Table 21 – Operations: CIM_ElementCapabilities**

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

710 **8.11 CIM_ElementCapabilities—References CIM_Processor and**
 711 **CIM_ProcessorCapabilities**

712 Table 22 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
 713 or shall not be supported.

714 **Table 22 – Operations: CIM_ElementCapabilities**

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

715 **8.12 CIM_ElementCapabilities—References CIM_ProcessorCore and**
 716 **CIM_EnabledLogicalElementCapabilities**

717 Table 23 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
 718 or shall not be supported.

719 **Table 23 – Operations: CIM_ElementCapabilities**

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

720 **8.13 CIM_EnabledLogicalElementCapabilities**

721 All operations in the default list in section 8.5 are supported as described by [DSP0200 version 1.2](#).

722 **8.14 CIM_HardwareThread**

723 Table 24 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
 724 or shall not be supported.

725 **Table 24 – Operations: CIM_HardwareThread**

Operation	Requirement	Messages
ModifyInstance	Optional. See section 8.14.1.	None

726 **8.14.1 CIM_HardwareThread—ModifyInstance**

727 This section details the requirements for the ModifyInstance operation applied to an instance of
 728 CIM_HardwareThread. The ModifyInstance operation may be supported.

729 The ModifyInstance operation shall be supported and the CIM_HardwareThread.ElementName property
 730 shall be modifiable when the ElementNameEditSupported property of the
 731 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_HardwareThread
 732 instance has a value of TRUE. See section 7.7.1.2.

733 **8.15 CIM_Memory**

734 Table 25 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
 735 or shall not be supported.

736 **Table 25 – Operations: CIM_Memory**

Operation	Requirement	Messages
ModifyInstance	Optional. See section 8.15.1.	None

737 **8.15.1 CIM_Memory—ModifyInstance**

738 This section details the requirements for the ModifyInstance operation applied to an instance of
739 CIM_Memory. The ModifyInstance operation may be supported.

740 The ModifyInstance operation shall be supported and the CIM_Memory.ElementName property shall be
741 modifiable when the ElementNameEditSupported property of the
742 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_Memory instance has
743 a value of TRUE. See section 7.8.1.2.

744 **8.16 CIM_Processor**

745 Table 26 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
746 or shall not be supported.

747 **Table 26 – Operations: CIM_Processor**

Operation	Requirement	Messages
ModifyInstance	Optional. See section 8.16.1.	None

748 **8.16.1 CIM_Processor—ModifyInstance**

749 This section details the requirements for the ModifyInstance operation applied to an instance of
750 CIM_Processor. The ModifyInstance operation may be supported.

751 The ModifyInstance operation shall be supported and the CIM_Processor.ElementName property shall be
752 modifiable when the ElementNameEditSupported property of the
753 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_Processor instance
754 has a value of TRUE. See section 7.2.4.

755 **8.17 CIM_ProcessorCapabilities**

756 All operations in the default list in section 8.5 are supported as described by [DSP0200 version 1.2](#).

757 **8.18 CIM_ProcessorCore**

758 Table 27 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
759 or shall not be supported.

760 **Table 27 – Operations: CIM_ProcessorCore**

Operation	Requirement	Messages
ModifyInstance	Optional. See section 8.18.1.	None

761 **8.18.1 CIM_ProcessorCore—ModifyInstance**

762 This section details the requirements for the ModifyInstance operation applied to an instance of
763 CIM_ProcessorCore. The ModifyInstance operation may be supported.

764 The ModifyInstance operation shall be supported and the CIM_ProcessorCore.ElementName property
765 shall be modifiable when the ElementNameEditSupported property of the
766 CIM_EnabledLogicalElementCapabilities instance that is associated with the CIM_ProcessorCore
767 instance has a value of TRUE. See section 7.6.1.2.

768 **8.19 CIM_SystemDevice**

769 Table 28 lists operations that either have special requirements beyond those from [DSP0200 version 1.2](#)
 770 or shall not be supported.

771 **Table 28 – Operations: CIM_SystemDevice**

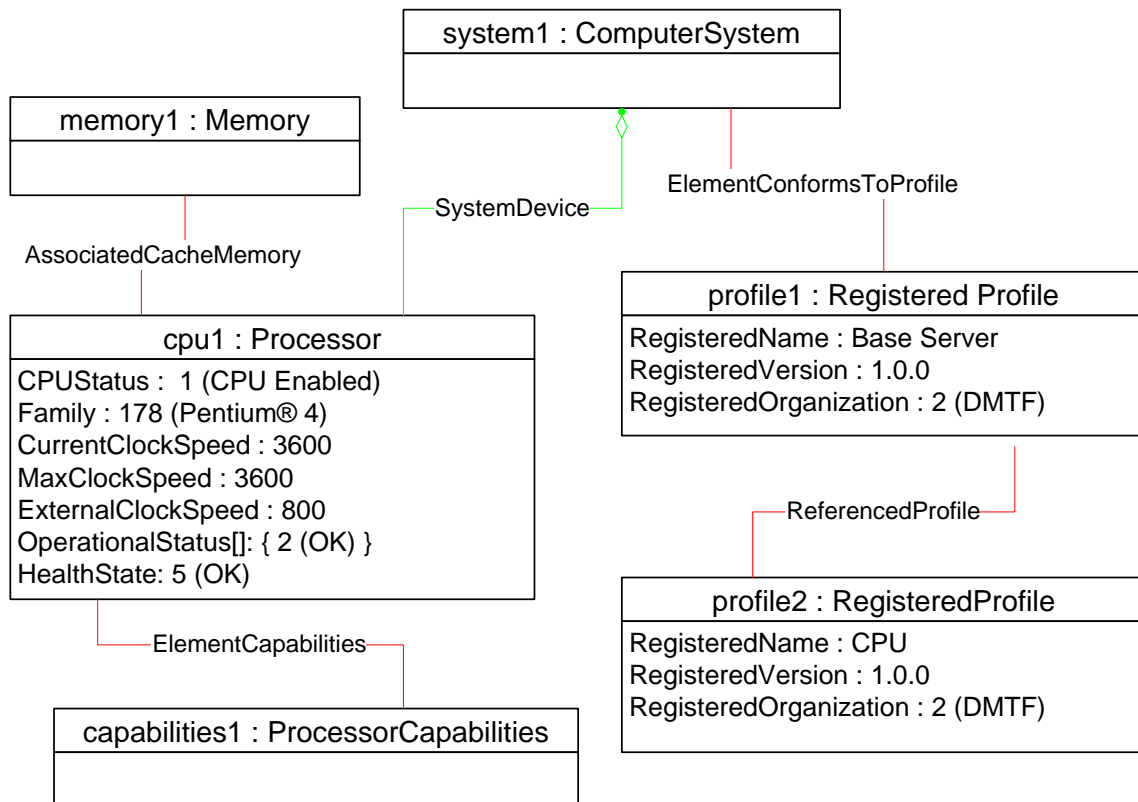
Operation	Requirement	Messages
Associators	Unspecified	None
AssociatorNames	Unspecified	None
References	Unspecified	None
ReferenceNames	Unspecified	None

772 **9 Use Cases**

773 This section contains object diagrams and use cases for the *CPU Profile*.

774 **9.1 Object Diagrams**

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

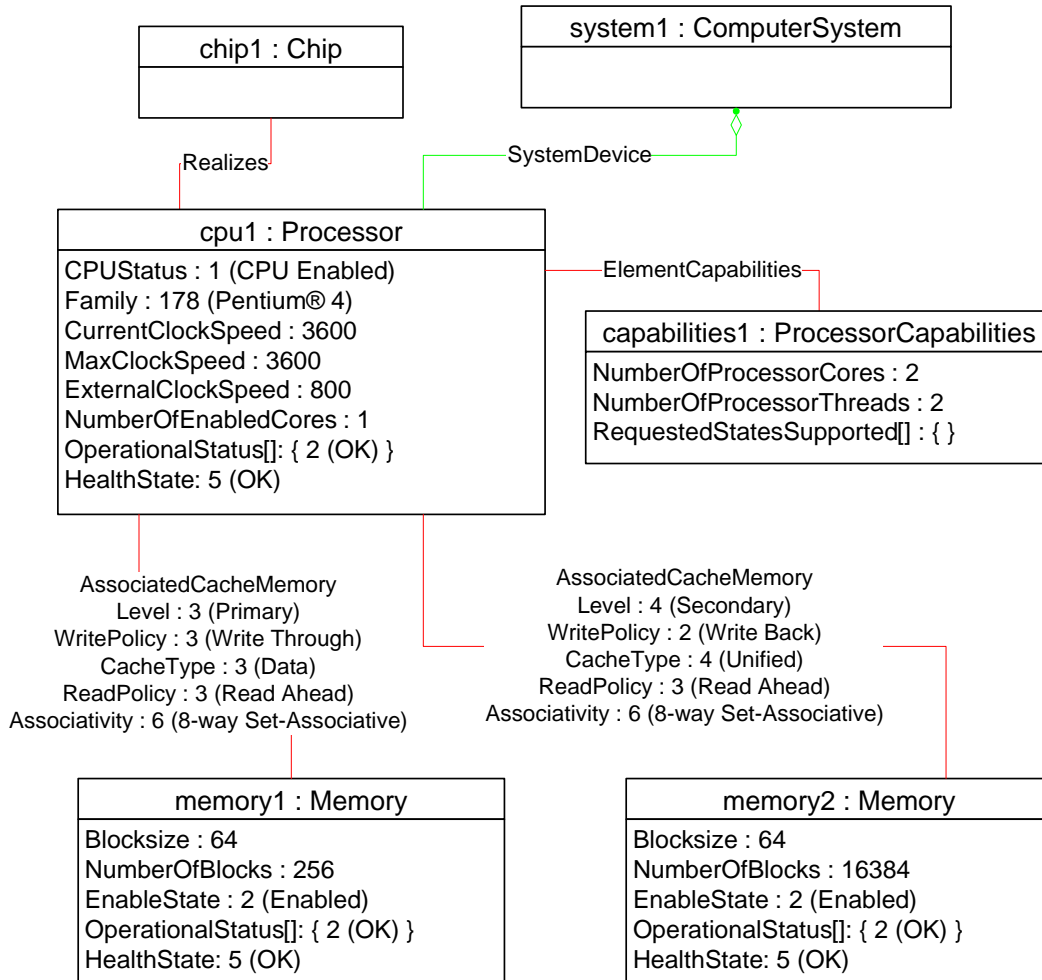


779

780

Figure 2 – Registered Profile

781 Figure 3 represents a possible instantiation of the *CPU Profile* representing a dual core processor, cpu1.
 782 The individual cores and hardware threads of cpu1 are not represented, but capabilities1 advertises that
 783 cpu1 is a dual core processor capable of two hardware threads, one thread per each core. If system1
 784 supports *SMBIOS Reference Specification 2.6* or later, the value of the NumberOfProcessorCores
 785 property will be equal to the SMBIOS Processor Information structure's Core Count structure value, and
 786 the value of the NumberOfHardwareThreads property will be equal to the SMBIOS Processor Information
 787 structure's Thread Count structure value. memory1 and memory2 are the cache memories of cpu1.
 788 Memory1 represents the level 1 cache, and memory2 is the level 2 cache, as denoted by the instances of
 789 CIM_AssociatedCacheMemory that associate memory1 and memory2 with cpu1. The physical aspects of
 790 cpu1 are represented by chip1, associated to cpu1 through an instance of CIM_Realizes.

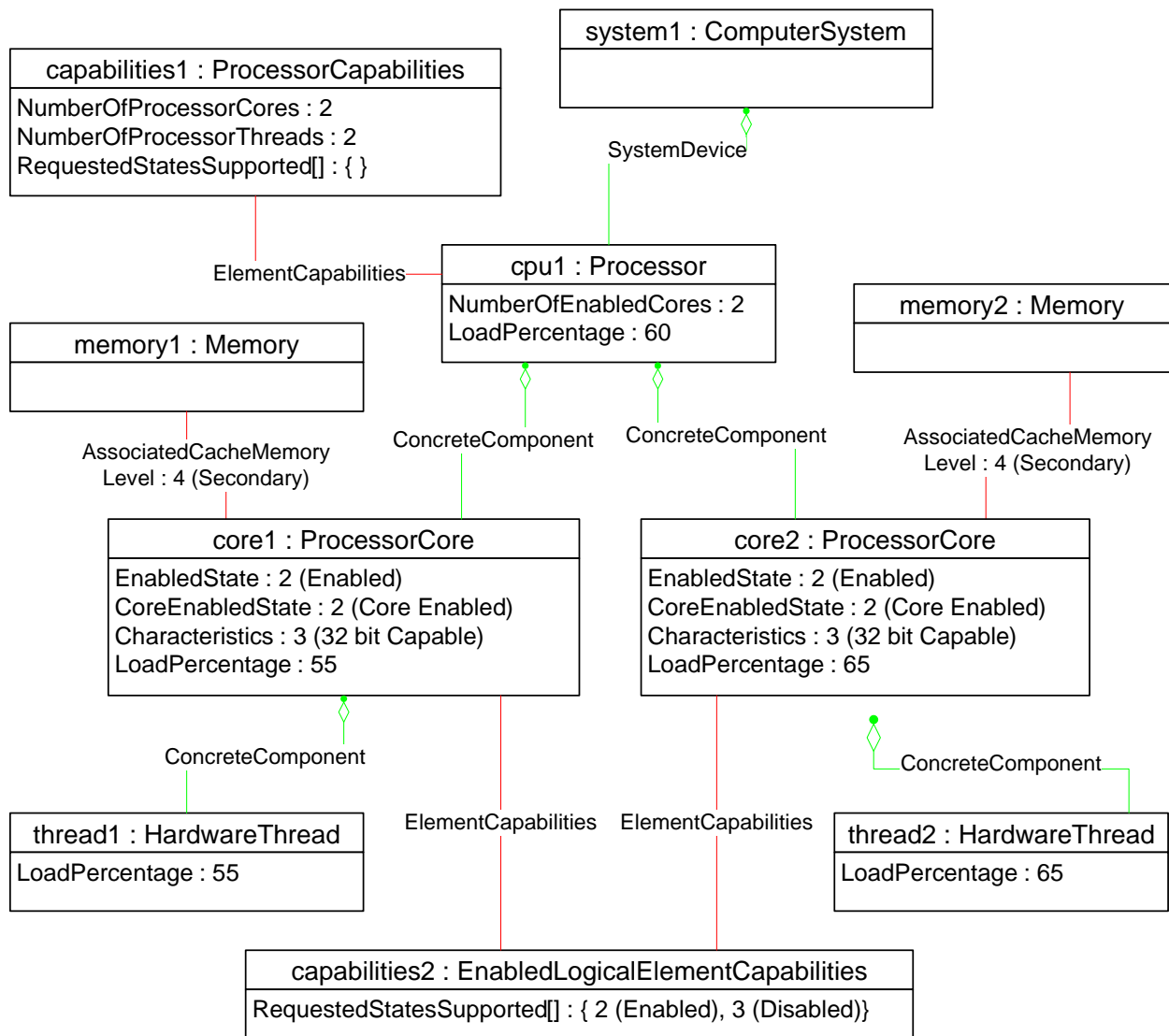


791

792

Figure 3 – Multi-core CPU

793 Figure 4 represents a possible instantiation of the *CPU Profile* representing a dual core processor, cpu1.
 794 Each of the processor cores is represented by an instance of CIM_ProcessorCore: core1 and core2,
 795 associated to the Host Processor, cpu1, through instances of CIM_ConcreteComponent. Each of the
 796 cores has one hardware thread, represented by thread1 and thread2, associated with it through instances
 797 of CIM_ConcreteComponent. The cache memories, memory1 and memory2, are associated to the
 798 processor cores that use them. Based on the capabilities of core1 and core2, represented by
 799 capabilities2, both processor cores can be enabled or disabled using the RequestStateChange() method.
 800 Figure 5 shows the same instantiation of *CPU Profile* after the RequestStateChange() method on core2
 801 has successfully executed.

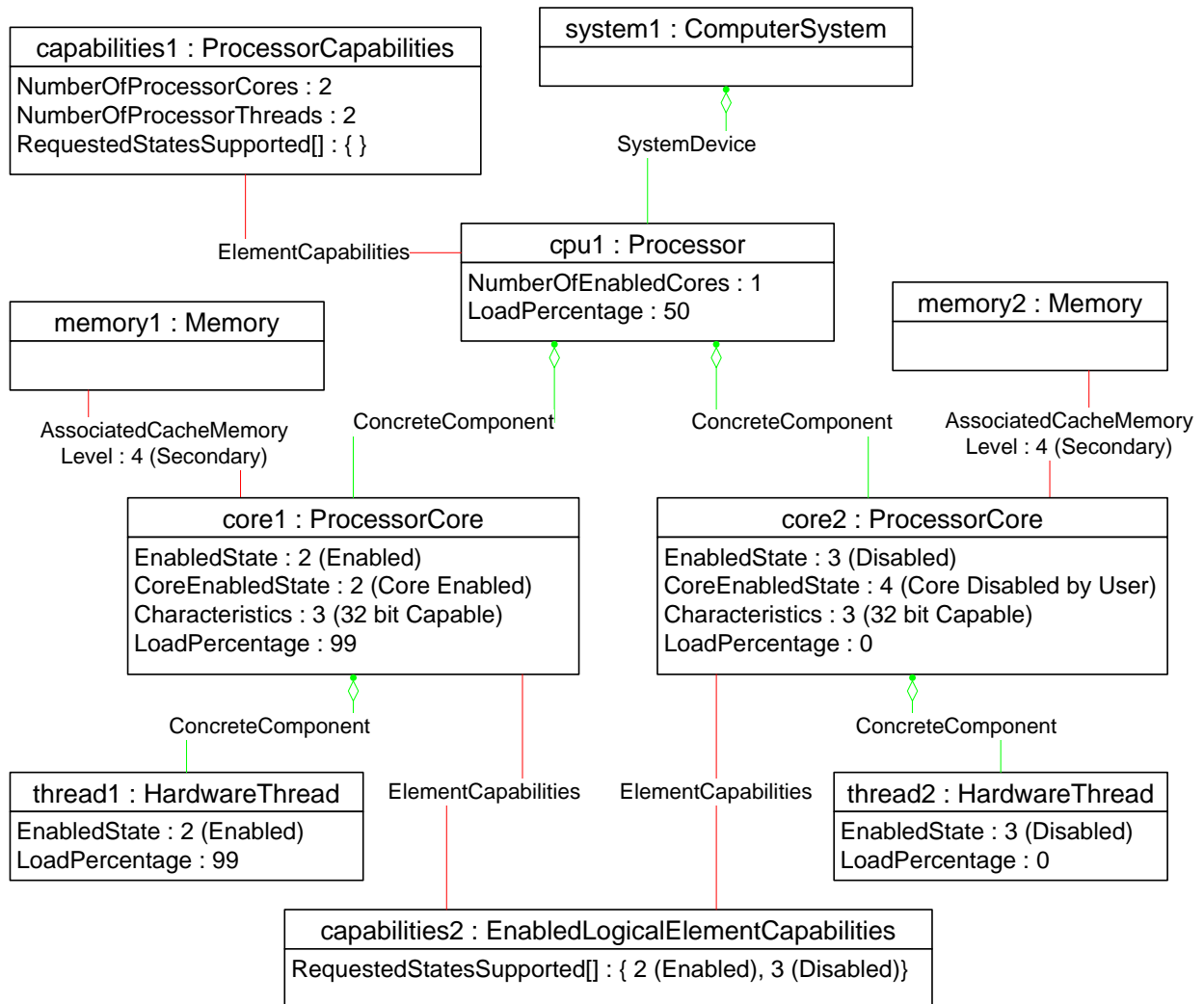


802

803

Figure 4 – Detailed Multi-core CPU

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



808

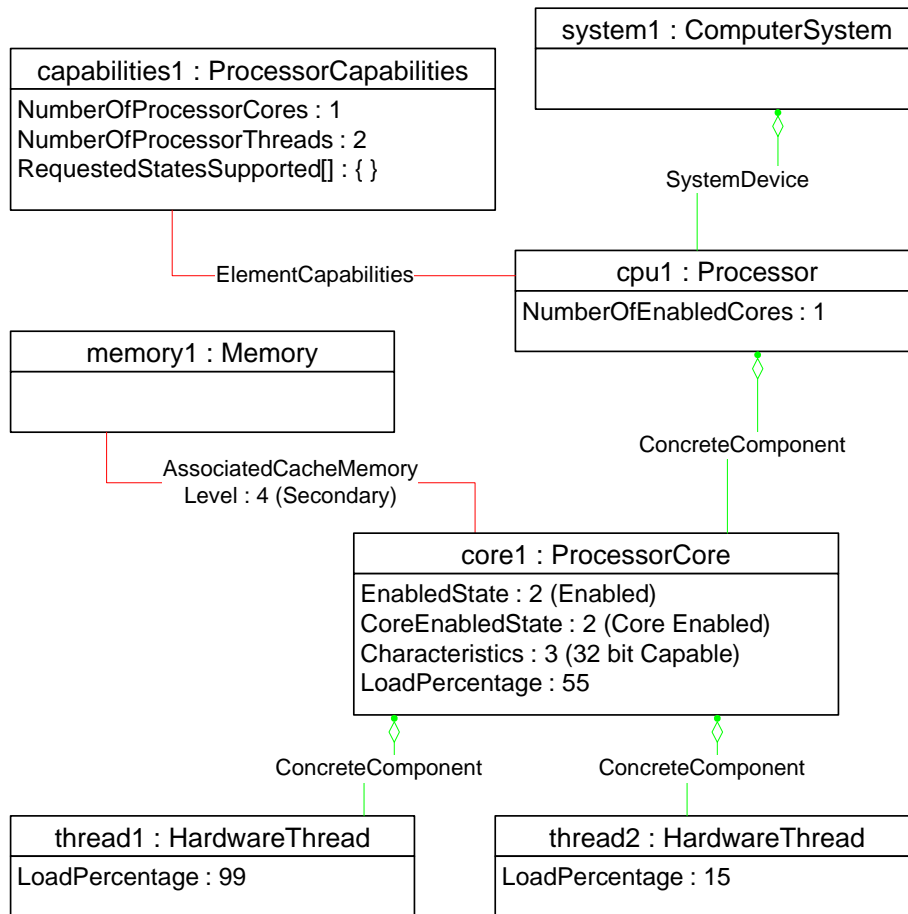
809 **Figure 5 – Multi-core CPU with a Disabled Core**

810 Figure 6 represents a possible instantiation of the *CPU Profile* representing a single core processor with
 811 multiple threads. thread1 and thread2 represent the hardware threads that exist on core1 and are
 812 associated to core1 through an instance of CIM_ConcreteComponent. cpu1 advertises the capabilities of
 813 multiple hardware threads through the capabilities1 NumberOfProcessorThreads property. The cache
 814 memory, memory1, is associated to core1, which is using the cache memory.

815 **EXPERIMENTAL**

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

818 **EXPERIMENTAL**

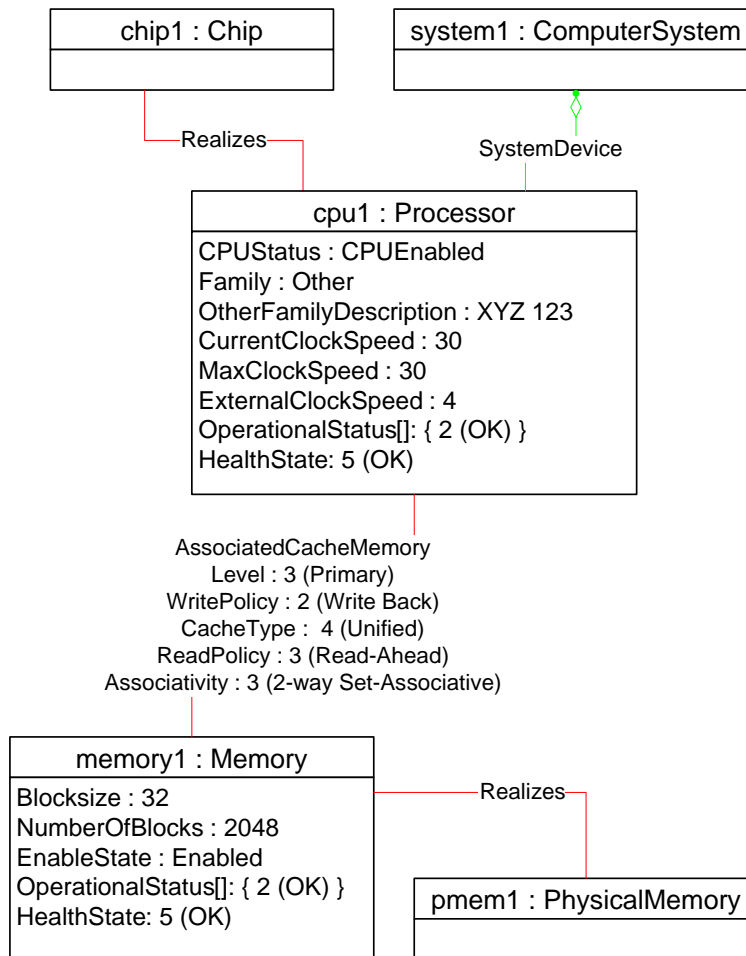


819

820

Figure 6 – Single Core, Multi-hardware Thread CPU

821 Figure 7 represents another instantiation of the *CPU Profile*. In this case, cpu1's cache memory,
 822 memory1, has a separate package represented by pmem1 and associated to memory1 through an
 823 instance of CIM_Realizes. The existence of pmem1 associated with the cpu1's cache memory indicates
 824 that the processor uses off-chip cache memory.



825

826

Figure 7 – Processor with Off-Chip Cache

827 **9.2 Change the Enabled State of the Memory to the Desired State**

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

- 829 1) Select the instance of CIM_Memory.
- 830 2) Select the associated instance of CIM_EnabledLogicalElementCapabilities and verify whether
 831 the RequestedStatesSupported property contains the desired state.
- 832 3) If the RequestedStatesSupported property contains the desired state, select the instance of
 833 CIM_Memory and execute the RequestStateChange() method with the desired state as a
 834 RequestedState parameter.

835 After the successful execution of the method, the EnabledState property of the instance of CIM_Memory
 836 will have the value of the desired state.

837 **9.3 Change the Enabled State of the CPU to the Desired State**

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

- 839 1. Select the instance of CIM_Processor.
- 840 2. Select the associated instance of CIM_ProcessorCapabilities and verify whether the
841 RequestedStatesSupported property contains the desired state.
- 842 3. If the RequestedStatesSupported property contains the desired state, select the instance of
843 CIM_Processor and execute the RequestStateChange() method with the desired state as a
844 RequestedState parameter.

845 After the successful execution of the method, the EnabledState property of the instance of
846 CIM_Processor will have the value of the desired state.

847 **9.4 Change the Enabled State of the CPU's Core to the Desired State**

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

- 849 1. Select the instance of CIM_ProcessorCore.
- 850 2. Select the associated instance of CIM_EnabledLogicalElementCapabilities and verify whether
851 the RequestedStatesSupported property contains the desired state.
- 852 3. If the RequestedStatesSupported property contains the desired state, select the instance of
853 CIM_ProcessorCore and execute the RequestStateChange() method with the desired state as
854 a RequestedState parameter.

855 After the successful execution of the method, the EnabledState property of the instance of
856 CIM_ProcessorCore will have the value of the desired state.

857 **9.5 Change the Enabled State of the CPU's Hardware Thread to the Desired 858 State**

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

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

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

868 **9.6 Retrieve All the Processor Cores for the CPU**

869 A client can retrieve all of the processor cores for the CPU by selecting all the CIM_ProcessorCore
870 instances that are associated with the given instance of CIM_Processor through instances of
871 CIM_Component.

872 **9.7 Retrieve All the Hardware Threads for the CPU**

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

- 874 1. Select all the CIM_ProcessorCore instances that are associated with the given instance of
875 CIM_Processor through instances of CIM_Component.

- 876 2. For each instance of CIM_ProcessorCore, select the instances of CIM_HardwareThread that
 877 are associated through instances of CIM_Component.

878 **9.8 Retrieve CPU’s Cache Memory Information for the CPU**

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

- 880 1. Select all the instances of CIM_ProcessorCore that are associated with the given instance of
 881 CIM_Processor through instances of CIM_Component.
- 882 2. If no instance of CIM_ProcessorCore exists, select the instances of
 883 CIM_AssociatedCacheMemory that reference the given instance of CIM_Processor, as well as
 884 all the instances of CIM_Memory that are associated with the given instance of CIM_Processor
 885 through instances of CIM_AssociatedCacheMemory.
- 886 3. Otherwise, for each instance of CIM_ProcessorCore, select the instances of
 887 CIM_AssociatedCacheMemory that reference the instance of CIM_ProcessorCore, as well as
 888 all the instances of CIM_Memory that are associated with the instance of CIM_ProcessorCore
 889 through instances of CIM_AssociatedCacheMemory.

890 **10 CIM Elements**

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

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

Element Name	Requirement	Description
Classes		
CIM_AssociatedCacheMemory	Optional	See sections 10.1 and 7.8.
CIM_ConcreteComponent (references CIM_HardwareThread and CIM_ProcessorCore)	Optional	See section 10.2.
CIM_ConcreteComponent (references CIM_ProcessorCore and CIM_Processor)	Optional	See section 10.3.
CIM_ElementCapabilities (references CIM_HardwareThread and CIM_EnabledLogicalElementCapabilities)	Optional	See section 10.4.
CIM_ElementCapabilities (references CIM_Memory and CIM_EnabledLogicalElementCapabilities)	Optional	See section 10.5.
CIM_ElementCapabilities (references CIM_Processor and CIM_ProcessorCapabilities)	Optional	See section 10.6.
CIM_ElementCapabilities (references CIM_ProcessorCore and CIM_EnabledLogicalElementCapabilities)	Optional	See section 10.7.
CIM_EnabledLogicalElementCapabilities	Optional	See sections 7.6.1, 7.7.1, 7.8.1, and 10.7.
CIM_HardwareThread	Optional	See section 10.9.

Element Name	Requirement	Description
CIM_Memory	Optional	See sections 10.10 and 7.8.
CIM_Processor	Mandatory	See sections 7.1 and 10.11.
CIM_ProcessorCapabilities	Optional	See sections 7.2 and 10.12.
CIM_ProcessorCore	Optional	See section 10.13.
CIM_RegisteredProfile	Mandatory	See section 10.14.
CIM_SystemDevice	Mandatory	See section 10.15.
Indications		
None defined in this profile		

895 10.1 CIM_AssociatedCacheMemory

896 CIM_AssociatedCacheMemory associates an instance of CIM_Processor or CIM_ProcessorCore with an
 897 instance of CIM_Memory that represents the cache memory of the processor. Table 30 contains the
 898 requirements for elements of this class.

899 **Table 30 – Class: CIM_AssociatedCacheMemory**

Elements	Requirement	Notes
Antecedent	Mandatory	Key: This property shall reference the instance of CIM_Memory that represents the cache memory.
Dependent	Mandatory	Key: This property shall reference the instance of CIM_Processor or CIM_ProcessorCore. See section 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).

900 10.2 CIM_ConcreteComponent—References CIM_HardwareThread and 901 CIM_ProcessorCore

902 CIM_ConcreteComponent associates an instance of CIM_ProcessorCore (the Threading Processor Core)
 903 with an instance CIM_HardwareThread that represents a hardware thread. CIM_ConcreteComponent
 904 shall be instantiated when the Threading Processor Core and the instance of CIM_HardwareThread are
 905 instantiated. Table 31 contains the requirements for elements of this class.

906 **Table 31 – Class: CIM_ConcreteComponent—References CIM_HardwareThread and**
 907 **CIM_ProcessorCore**

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

908 **10.3 CIM_ConcreteComponent—References CIM_ProcessorCore and**
 909 **CIM_Processor**

910 CIM_ConcreteComponent associates an instance of CIM_Processor (the Host Processor) with an
 911 instance CIM_ProcessorCore that represents a processor core. CIM_ConcreteComponent shall be
 912 instantiated when the Host Processor and the instance of CIM_ProcessorCore are instantiated. Table 32
 913 contains the requirements for elements of this class.

914 **Table 32 – Class: CIM_ConcreteComponent—References CIM_ProcessorCore and CIM_Processor**

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

915 **10.4 CIM_ElementCapabilities—References CIM_HardwareThread and**
 916 **CIM_EnabledLogicalElementCapabilities**

917 CIM_ElementCapabilities associates an instance of CIM_HardwareThread with the instance of
 918 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
 919 CIM_HardwareThread.

920 CIM_ElementCapabilities is mandatory when the instance of CIM_HardwareThread and the instance of
 921 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
 922 CIM_HardwareThread exist. Table 33 contains the requirements for elements of this class.

923 **Table 33 – Class: CIM_ElementCapabilities—References CIM_HardwareThread and**
 924 **CIM_EnabledLogicalElementCapabilities**

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

925 **10.5 CIM_ElementCapabilities—References CIM_Memory and**
 926 **CIM_EnabledLogicalElementCapabilities**

927 CIM_ElementCapabilities associates an instance of CIM_Memory with the instance of
 928 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of CIM_Memory.

929 CIM_ElementCapabilities is mandatory when the instance of CIM_Memory and the instance of
 930 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of CIM_Memory
 931 exist. Table 34 contains the requirements for elements of this class.

932 **Table 34 – Class: CIM_ElementCapabilities—References CIM_Memory and**
 933 **CIM_EnabledLogicalElementCapabilities**

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

934 **10.6 CIM_ElementCapabilities—References CIM_Processor and**
 935 **CIM_ProcessorCapabilities**

936 CIM_ElementCapabilities associates an instance of CIM_Processor with the instance of
 937 CIM_ProcessorCapabilities that describes the capabilities of the instance of CIM_Processor.

938 CIM_ElementCapabilities is mandatory when the instance of CIM_Processor and the instance of
 939 CIM_ProcessorCapabilities exist. Table 35 contains the requirements for elements of this class.

940 **Table 35 – Class: CIM_ElementCapabilities—References CIM_Processor and**
 941 **CIM_ProcessorCapabilities**

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

942 **10.7 CIM_ElementCapabilities—References CIM_ProcessorCore and**
 943 **CIM_EnabledLogicalElementCapabilities**

944 CIM_ElementCapabilities associates an instance of CIM_ProcessorCore with the instance of
 945 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
 946 CIM_ProcessorCore.

947 CIM_ElementCapabilities is mandatory when the instance of CIM_ProcessorCore and the instance of
 948 CIM_EnabledLogicalElementCapabilities that describes the capabilities of the instance of
 949 CIM_ProcessorCore exist. Table 36 contains the requirements for elements of this class.

950 **Table 36 – Class: CIM_ElementCapabilities—References CIM_ProcessorCore and**
 951 **CIM_EnabledLogicalElementCapabilities**

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

952 **10.8 CIM_EnabledLogicalElementCapabilities**

953 CIM_EnabledLogicalElementCapabilities represents the capabilities of the memory, the processor core,
 954 or the hardware thread. Table 37 contains the requirements for elements of this class.

955 **Table 37 – Class: CIM_EnabledLogicalElementCapabilities**

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

956 **10.9 CIM_HardwareThread**

957 CIM_HardwareThread represents the hardware thread of the processor. Table 38 contains the
 958 requirements for elements of this class.

959 **Table 38 – Class: CIM_HardwareThread**

Elements	Requirement	Notes
InstanceID	Mandatory	Key
LoadPercentage	Optional	EXPERIMENTAL
EnabledState	Mandatory	See section 7.7.4.
RequestedState	Mandatory	See section 7.7.3.
OperationalStatus	Mandatory	None
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern “.*”.
RequestStateChange()	Conditional	See section 8.3.

960 **10.10 CIM_Memory**

961 CIM_Memory represents the CPU’s cache memory. Table 39 contains the requirements for elements of
 962 this class.

963 **Table 39 – Class: CIM_Memory**

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

964 **10.11 CIM_Processor**

965 CIM_Processor represents the processor or CPU. Table 40 contains the requirements for elements of this
 966 class.

967 **Table 40 – Class: CIM_Processor**

Elements	Requirement	Notes
SystemCreationClassName	Mandatory	Key
SystemName	Mandatory	Key
CreationClassName	Mandatory	Key
DeviceID	Mandatory	Key
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 section 7.5.1.
LoadPercentage	Optional	None
EnabledState	Mandatory	See section 7.5.2.
RequestedState	Mandatory	See section 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 section 8.1.

968 **10.12 CIM_ProcessorCapabilities**

969 CIM_ProcessorCapabilities represents the capabilities of the processor. Table 41 contains the
 970 requirements for elements of this class.

971 **Table 41 – Class: CIM_ProcessorCapabilities**

Elements	Requirement	Notes
InstanceID	Mandatory	Key
NumberOfProcessorCores	Mandatory	A value of 0 shall mean “Unknown”.
NumberOfHardwareThreads	Mandatory	A value of 0 shall mean “Unknown”.
RequestedStatesSupported	Mandatory	See section 7.2.2.
ElementNameEditSupported	Mandatory	See section 7.2.4.
MaxElementNameLen	Conditional	See section 7.2.5.

972 **10.13 CIM_ProcessorCore**

973 CIM_ProcessorCore represents the core of the processor. Table 42 contains the requirements for
 974 elements of this class.

975 **Table 42 – Class: CIM_ProcessorCore**

Elements	Requirement	Notes
InstanceID	Mandatory	Key
CoreEnabledState	Mandatory	See section 7.6.4.1.
LoadPercentage	Optional	EXPERIMENTAL
EnabledState	Mandatory	See section 7.6.4.2.
RequestedState	Mandatory	See section 7.6.3.
OperationalStatus	Mandatory	None
HealthState	Mandatory	None
ElementName	Mandatory	The property shall match the pattern ".*".
RequestStateChange()	Conditional	See section 8.2.

976 **10.14 CIM_RegisteredProfile**

977 The CIM_RegisteredProfile class is defined by the *Profile Registration Profile*. The requirements denoted
 978 in Table 43 are in addition to those mandated by the *Profile Registration Profile*.

979 **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 "1.0.0".
RegisteredOrganization	Mandatory	This property shall have a value of 2 (DMTF).

980 NOTE: Previous versions of this document included the suffix "Profile" for the RegisteredName value. If
 981 implementations querying for the RegisteredName value find the suffix "Profile", they should ignore the suffix, with
 982 any surrounding white spaces, before any comparison is done with the value as specified in this document.

983 **10.15 CIM_SystemDevice**

984 CIM_SystemDevice associates an instance of CIM_Processor with the instance of CIM_ComputerSystem
 985 of which the CIM_Processor instance is a member. Table 44 contains the requirements for elements of
 986 this class.

987 **Table 44 – Class: CIM_SystemDevice**

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

988

989
990
991
992

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

993

994
995
996
997

ANNEX B (informative)

Acknowledgments

998 The authors wish to acknowledge the following people.

999 **Editors:**

- 1000 • Jon Hass – Dell
- 1001 • Khachatur Papanyan – Dell
- 1002 • Jeff Hilland – HP

1003 **Contributors:**

- 1004 • Jon Hass – Dell
- 1005 • Khachatur Papanyan – Dell
- 1006 • Jeff Hilland – HP
- 1007 • Christina Shaw – HP
- 1008 • Aaron Merkin – IBM
- 1009 • Jeff Lynch – IBM
- 1010 • Perry Vincent – Intel
- 1011 • John Leung – Intel
- 1012