



1  
2  
3  
4

**Document Number: DSP0808**

**Date: 2009-06-04**

**Version: 1.0.0**

# 5 **CPU Profile SM CLP Mapping Specification**

6 **Document Type: Specification**  
7 **Document Status: DMTF Standard**  
8 **Document Language: E**  
9

10 Copyright notice

11 Copyright © 2006, 2009 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

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

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26 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

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

32

# CONTENTS

33 Foreword ..... 5  
 34 Introduction ..... 6  
 35 1 Scope ..... 7  
 36 2 Normative References..... 7  
 37 2.1 Approved References ..... 7  
 38 2.2 Other References..... 7  
 39 3 Terms and Definitions..... 7  
 40 4 Symbols and Abbreviated Terms..... 8  
 41 5 Recipes..... 9  
 42 6 Mappings..... 9  
 43 6.1 CIM\_AssociatedCacheMemory ..... 9  
 44 6.2 CIM\_ConcreteComponent ..... 12  
 45 6.3 CIM\_HardwareThread ..... 15  
 46 6.4 CIM\_Memory ..... 20  
 47 6.5 CIM\_ElementCapabilities ..... 24  
 48 6.6 CIM\_EnabledLogicalElementCapabilities..... 30  
 49 6.7 CIM\_Processor ..... 32  
 50 6.8 CIM\_ProcessorCapabilities ..... 36  
 51 6.9 CIM\_ProcessorCore ..... 38  
 52 6.10 CIM\_SystemDevice ..... 42  
 53 ANNEX A (informative) Change Log ..... 46  
 54

## 55 Tables

56 Table 1 – Command Verb Requirements for CIM\_AssociatedCacheMemory ..... 9  
 57 Table 2 – Command Verb Requirements for CIM\_ConcreteComponent ..... 12  
 58 Table 3 – Command Verb Requirements for CIM\_HardwareThread ..... 15  
 59 Table 4 – Command Verb Requirements for CIM\_Memory ..... 20  
 60 Table 5 – Command Verb Requirements for CIM\_ElementCapabilities ..... 24  
 61 Table 6 – Command Verb Requirements for CIM\_EnabledLogicalElementCapabilities..... 30  
 62 Table 7 – Command Verb Requirements for CIM\_Processor ..... 32  
 63 Table 8 – Command Verb Requirements for CIM\_ProcessorCapabilities ..... 36  
 64 Table 9 – Command Verb Requirements for CIM\_ProcessorCore ..... 38  
 65 Table 10 – Command Verb Requirements for CIM\_SystemDevice ..... 43  
 66



68

## Foreword

69 The *CPU Profile SM CLP Mapping Specification* (DSP0808) was prepared by the Server Management  
70 Working Group.

### 71 **Conventions**

72 The pseudo-code conventions utilized in this document are the Recipe Conventions as defined in SNIA  
73 [SMI-S 1.1.0](#), section 7.6.

### 74 **Acknowledgements**

75 The authors wish to acknowledge the following participants from the DMTF Server Management Working  
76 Group:

- 77 • Khachatur Papanyan – Dell Inc.
- 78 • Jon Hass – Dell Inc.
- 79 • Jeff Hilland – HP
- 80 • Christina Shaw – HP
- 81 • Aaron Merkin – IBM
- 82 • Jeff Lynch – IBM
- 83 • Perry Vincent – Intel
- 84 • John Leung – Intel.

85

86

## Introduction

87 This document defines the SM CLP mapping for CIM elements described in the [CPU Profile](#). The  
88 information in this specification, combined with the [SM CLP-to-CIM Common Mapping Specification 1.0](#),  
89 is intended to be sufficient to implement SM CLP commands relevant to the classes, properties, and  
90 methods described in the [CPU Profile](#) using CIM operations.

91 The target audience for this specification is implementers of the SM CLP support for the [CPU Profile](#).

92

# CPU Profile SM CLP Mapping Specification

## 93 1 Scope

94 This specification contains the requirements for an implementation of the SM CLP to provide access to,  
95 and implement the behaviors of, the [CPU Profile](#).

## 96 2 Normative References

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

### 100 2.1 Approved References

101 DMTF DSP1022, *CPU Profile 1.0*,  
102 [http://www.dmtf.org/standards/published\\_documents/DSP1022\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1022_1.0.pdf)

103 DMTF DSP0216, *SM CLP-to-CIM Common Mapping Specification 1.0*,  
104 [http://www.dmtf.org/standards/published\\_documents/DSP0216\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP0216_1.0.pdf)

105 SNIA, *Storage Management Initiative Specification (SMI-S) 1.1.0*,  
106 [http://www.snia.org/tech\\_activities/standards/curr\\_standards/smi](http://www.snia.org/tech_activities/standards/curr_standards/smi)

### 107 2.2 Other References

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

## 110 3 Terms and Definitions

111 For the purposes of this document, the following terms and definitions apply.

### 112 3.1

#### 113 **can**

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

### 115 3.2

#### 116 **cannot**

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

### 118 3.3

#### 119 **conditional**

120 indicates requirements to be followed strictly in order to conform to the document when the specified  
121 conditions are met

### 122 3.4

#### 123 **mandatory**

124 indicates requirements to be followed strictly in order to conform to the document and from which no  
125 deviation is permitted

- 126 **3.5**  
127 **may**  
128 indicates a course of action permissible within the limits of the document
- 129 **3.6**  
130 **need not**  
131 indicates a course of action permissible within the limits of the document
- 132 **3.7**  
133 **optional**  
134 indicates a course of action permissible within the limits of the document
- 135 **3.8**  
136 **shall**  
137 indicates requirements to be followed strictly in order to conform to the document and from which no  
138 deviation is permitted
- 139 **3.9**  
140 **shall not**  
141 indicates requirements to be followed strictly in order to conform to the document and from which no  
142 deviation is permitted
- 143 **3.10**  
144 **should**  
145 indicates that among several possibilities, one is recommended as particularly suitable, without  
146 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 147 **3.11**  
148 **should not**  
149 indicates that a certain possibility or course of action is deprecated but not prohibited

## 150 **4 Symbols and Abbreviated Terms**

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

- 152 **4.1**  
153 **CIM**  
154 Common Information Model
- 155 **4.2**  
156 **CLP**  
157 Command Line Protocol
- 158 **4.3**  
159 **DMTF**  
160 Distributed Management Task Force
- 161 **4.4**  
162 **IETF**  
163 Internet Engineering Task Force



- 164 **4.5**  
 165 **SM**  
 166 Server Management
- 167 **4.6**  
 168 **SMI-S**  
 169 Storage Management Initiative Specification
- 170 **4.7**  
 171 **SNIA**  
 172 Storage Networking Industry Association

## 173 **5 Recipes**

174 The following is a list of the common recipes used by the mappings in this specification. For a definition of  
 175 each recipe, see the *SM CLP-to-CIM Common Mapping Specification 1.0* ([DSP0216](#)).

- 176 • smResetRSC
- 177 • smShowInstance
- 178 • smShowInstances
- 179 • smShowAssociationInstance
- 180 • smShowAssociationInstances
- 181 • smStartRSC
- 182 • smStopRSC

## 183 **6 Mappings**

184 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in  
 185 the [CPU Profile](#). Requirements specified here related to support for a CLP verb for a particular class are  
 186 solely within the context of this profile.

### 187 **6.1 CIM\_AssociatedCacheMemory**

188 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

189 Table 1 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 190 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 191 target. Table 1 is for informational purposes only; in case of a conflict between Table 1 and requirements  
 192 detailed in the following sections, the text detailed in the following sections supersedes the information in  
 193 Table 1.

194 **Table 1 – Command Verb Requirements for CIM\_AssociatedCacheMemory**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	

Command Verb	Requirement	Comments
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.1.2
Start	Not supported	
Stop	Not supported	

195 No mapping is defined for the following verbs for the specified target: create, delete, dump, load,  
196 reset, set, start, and stop.

### 197 6.1.1 Ordering of Results

198 When results are returned for multiple instances of CIM\_AssociatedCacheMemory, implementations shall  
199 utilize the following algorithm to produce the natural (that is, default) ordering:

- 200 • Results for CIM\_AssociatedCacheMemory are unordered; therefore, no algorithm is defined.

### 201 6.1.2 Show

202 This section describes how to implement the `show` verb when applied to an instance of  
203 CIM\_AssociatedCacheMemory. Implementations shall support the use of the `show` verb with  
204 CIM\_AssociatedCacheMemory.

#### 205 6.1.2.1 Show Command Form for Multiple Instances Target – CIM\_Processor Reference

206 This command form is used to show many instances of CIM\_AssociatedCacheMemory. This command  
207 form corresponds to a `show` command issued against the instance of CIM\_AssociatedCacheMemory  
208 where only one reference is specified and the reference is to the instance of CIM\_Processor.

##### 209 6.1.2.1.1 Command Form

```
210 show <CIM_AssociatedCacheMemory multiple instances>
```

##### 211 6.1.2.1.2 CIM Requirements

212 See CIM\_AssociatedCacheMemory in the “CIM Elements” section of the [CPU Profile](#) for the list of  
213 mandatory properties.

##### 214 6.1.2.1.3 Behavior Requirements

###### 215 6.1.2.1.3.1 Preconditions

216 In this section `$instance` represents the instance of a CIM\_Processor, which is referenced by  
217 CIM\_AssociatedCacheMemory.

###### 218 6.1.2.1.3.2 Pseudo Code

```
219 &smShowAssociationInstances ( "CIM_AssociatedCacheMemory",  
220     $instance.GetObjectPath() );  
221 &smEnd;
```

#### 222 6.1.2.2 Show Command Form for Multiple Instances Target – CIM\_ProcessorCore Reference

223 This command form is used to show many instances of CIM\_AssociatedCacheMemory. This command  
224 form corresponds to a `show` command issued against the instance of CIM\_AssociatedCacheMemory  
225 where only one reference is specified and the reference is to the instance of CIM\_ProcessorCore.

### 226 6.1.2.2.1 Command Form

```
227 show <CIM_AssociatedCacheMemory multiple instances>
```

### 228 6.1.2.2.2 CIM Requirements

229 See CIM\_AssociatedCacheMemory in the “CIM Elements” section of the [CPU Profile](#) for the list of  
230 mandatory properties.

### 231 6.1.2.2.3 Behavior Requirements

#### 232 6.1.2.2.3.1 Preconditions

233 In this section \$instance represents the instance of a CIM\_ProcessorCore, which is referenced by  
234 CIM\_AssociatedCacheMemory.

#### 235 6.1.2.2.3.2 Pseudo Code

```
236 &smShowAssociationInstances ( "CIM_AssociatedCacheMemory",  
237     $instance.getObjectPath() );  
238 &smEnd;
```

### 239 6.1.2.3 Show Command Form for Multiple Instances Target – CIM\_Memory Reference

240 This command form is used to show many instances of CIM\_AssociatedCacheMemory. This command  
241 form corresponds to a show command issued against the instance of CIM\_AssociatedCacheMemory  
242 where only one reference is specified and the reference is to the instance of CIM\_Memory.

#### 243 6.1.2.3.1 Command Form

```
244 show <CIM_AssociatedCacheMemory multiple instances>
```

#### 245 6.1.2.3.2 CIM Requirements

246 See CIM\_AssociatedCacheMemory in the “CIM Elements” section of the [CPU Profile](#) for the list of  
247 mandatory properties.

#### 248 6.1.2.3.3 Behavior Requirements

##### 249 6.1.2.3.3.1 Preconditions

250 In this section \$instance represents the instance of a CIM\_Memory, which is referenced by  
251 CIM\_AssociatedCacheMemory.

##### 252 6.1.2.3.3.2 Pseudo Code

```
253 &smShowAssociationInstances ( "CIM_AssociatedCacheMemory",  
254     $instance.getObjectPath() );  
255 &smEnd;
```

### 256 6.1.2.4 Show Command Form for a Single Instance Target – Both References

257 This command form is for the show verb applied to a single instance. This command form corresponds to  
258 show command issued against CIM\_AssociatedCacheMemory where both references are specified and  
259 therefore the desired instance is unambiguously identified.

#### 260 6.1.2.4.1 Command Form

```
261 show <CIM_AssociatedCacheMemory single instance>
```

262 **6.1.2.4.1.1 CIM Requirements**

263 See CIM\_AssociatedCacheMemory in the “CIM Elements” section of the [CPU Profile](#) for the list of  
264 mandatory properties.

265 **6.1.2.4.2 Behavior Requirements**266 **6.1.2.4.2.1 Preconditions**

267 In this section \$instanceA represents the referenced instance of CIM\_Processor or  
268 CIM\_ProcessorCore through CIM\_AssociatedCacheMemory association. \$instanceB represents the  
269 instance of CIM\_Memory which is referenced by CIM\_AssociatedCacheMemory.

270 **6.1.2.4.2.2 Pseudo Code**

```
271 &smShowAssociationInstance ( "CIM_AssociatedCacheMemory", $instanceA.getObjectPath(),
272     $instanceB.getObjectPath() );
273 &smEnd;
```

274 **6.2 CIM\_ConcreteComponent**

275 The cd, exit, help, and version verbs shall be supported as described in [DSP0216](#).

276 Table 2 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
277 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
278 target. Table 2 is for informational purposes only; in case of a conflict between Table 2 and requirements  
279 detailed in the following sections, the text detailed in the following sections supersedes the information in  
280 Table 2.

281 **Table 2 – Command Verb Requirements for CIM\_ConcreteComponent**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.2.2.
Start	Not supported	
Stop	Not supported	

282 No mapping is defined for the following verbs for the specified target: create, delete, dump, load,  
283 reset, set, start, and stop.

284 **6.2.1 Ordering of Results**

285 When results are returned for multiple instances of CIM\_ConcreteComponent, implementations shall  
286 utilize the following algorithm to produce the natural (that is, default) ordering:

- 287 • Results for CIM\_ConcreteComponent are unordered; therefore, no algorithm is defined.

## 288 6.2.2 Show

289 This section describes how to implement the `show` verb when applied to an instance of  
 290 `CIM_ConcreteComponent`. Implementations shall support the use of the `show` verb with  
 291 `CIM_ConcreteComponent`.

### 292 6.2.2.1 Show Command Form for Multiple Instances Target – `CIM_Processor` Reference

293 This command form is used to show many instances of `CIM_ConcreteComponent`. This command form  
 294 corresponds to a `show` command issued against the instance of `CIM_ConcreteComponent` where only  
 295 one reference is specified and the reference is to the instance of `CIM_Processor`.

#### 296 6.2.2.1.1 Command Form

```
297 show <CIM_ConcreteComponent multiple instances>
```

#### 298 6.2.2.1.2 CIM Requirements

299 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
 300 properties.

#### 301 6.2.2.1.3 Behavior Requirements

##### 302 6.2.2.1.3.1 Preconditions

303 In this section `$instance` represents the instance of a `CIM_Processor`, which is referenced by  
 304 `CIM_ConcreteComponent`.

##### 305 6.2.2.1.3.2 Pseudo Code

```
306 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );  
307 &smEnd;
```

### 308 6.2.2.2 Show Command Form for Multiple Instances Target – `CIM_ProcessorCore` Reference

309 This command form is used to show many instances of `CIM_ConcreteComponent`. This command form  
 310 corresponds to a `show` command issued against the instance of `CIM_ConcreteComponent` where only  
 311 one reference is specified and the reference is to the instance of `CIM_ProcessorCore`.

#### 312 6.2.2.2.1 Command Form

```
313 show <CIM_ConcreteComponent multiple instances>
```

#### 314 6.2.2.2.2 CIM Requirements

315 See `CIM_AssociatedCacheMemory` in the “CIM Elements” section of the [CPU Profile](#) for the list of  
 316 mandatory properties.

#### 317 6.2.2.2.3 Behavior Requirements

##### 318 6.2.2.2.3.1 Preconditions

319 In this section `$instance` represents the instance of a `CIM_ProcessorCore`, which is referenced by  
 320 `CIM_ConcreteComponent`.

##### 321 6.2.2.2.3.2 Pseudo Code

```
322 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );  
323 &smEnd;
```

### 324 6.2.2.3 Show Command Form for a Single Instance Target – CIM\_HardwareThread Reference

325 This command form is used to show a single instance of CIM\_ConcreteComponent. This command form  
326 corresponds to a `show` command issued against a single instance of CIM\_ConcreteComponent, where  
327 only one reference is specified and the reference is to the instance of CIM\_HardwareThread.

#### 328 6.2.2.3.1 Command Form

```
329 show <CIM_ConcreteComponent single instance>
```

#### 330 6.2.2.3.2 CIM Requirements

331 See CIM\_AssociatedCacheMemory in the “CIM Elements” section of the [CPU Profile](#) for the list of  
332 mandatory properties.

#### 333 6.2.2.3.3 Behavior Requirements

##### 334 6.2.2.3.3.1 Preconditions

335 In this section `$instance` represents the instance of CIM\_HardwareThread which is referenced by  
336 CIM\_ConcreteComponent.

##### 337 6.2.2.3.3.2 Pseudo Code

```
338 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );  
339 &smEnd;
```

### 340 6.2.2.4 Show Command Form for a Single Instance Target – Both References: CIM\_Processor 341 and CIM\_ProcessorCore

342 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
343 a `show` command issued against CIM\_ConcreteComponent where both references are specified and  
344 therefore the desired instance is unambiguously identified.

#### 345 6.2.2.4.1 Command Form

```
346 show <CIM_ConcreteComponent single instance>
```

#### 347 6.2.2.4.2 CIM Requirements

348 See CIM\_AssociatedCacheMemory in the “CIM Elements” section of the [CPU Profile](#) for the list of  
349 mandatory properties.

#### 350 6.2.2.4.3 Behavior Requirements

##### 351 6.2.2.4.3.1 Preconditions

352 In this section `$instanceA` represents the referenced instance of CIM\_Processor through  
353 CIM\_ConcreteComponent association. `$instanceB` represents the instance of CIM\_ProcessorCore  
354 which is referenced by CIM\_ConcreteComponent.

##### 355 6.2.2.4.3.2 Pseudo Code

```
356 &smShowAssociationInstance ( "CIM_ConcreteComponent", $instanceA.getObjectPath(),  
357     $instanceB.getObjectPath() );  
358 &smEnd;
```

359 **6.2.2.5 Show Command Form for a Single Instance Target – Both References: CIM\_Processor**  
 360 **and CIM\_ProcessorCore**

361 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 362 a `show` command issued against `CIM_ConcreteComponent` where both references are specified and  
 363 therefore the desired instance is unambiguously identified.

364 **6.2.2.5.1 Command Form**

```
365 show <CIM_ConcreteComponent single instance>
```

366 **6.2.2.5.2 CIM Requirements**

367 See `CIM_AssociatedCacheMemory` in the “CIM Elements” section of the [CPU Profile](#) for the list of  
 368 mandatory properties.

369 **6.2.2.5.3 Behavior Requirements**

370 **6.2.2.5.3.1 Preconditions**

371 In this section `$instanceA` represents the referenced instance of `CIM_Processor` through  
 372 `CIM_ConcreteComponent` association. `$instanceB` represents the instance of `CIM_ProcessorCore`  
 373 which is referenced by `CIM_ConcreteComponent`.

374 **6.2.2.5.3.2 Pseudo Code**

```
375 &smShowAssociationInstance ( "CIM_ConcreteComponent", $instanceA.getObjectPath(),  

    376     $instanceB.getObjectPath() );  

    377 &smEnd;
```

378 **6.3 CIM\_HardwareThread**

379 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

380 Table 3 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 381 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 382 target. Table 3 is for informational purposes only; in case of a conflict between Table 3 and requirements  
 383 detailed in the following sections, the text detailed in the following sections supersedes the information in  
 384 Table 3.

385 **Table 3 – Command Verb Requirements for CIM\_HardwareThread**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.3.2.
Set	May	See 6.3.3.
Show	Shall	See 6.3.4.
Start	May	See 6.3.5.
Stop	May	See 6.3.6.

386 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, and `load`.

### 387 6.3.1 Ordering of Results

388 When results are returned for multiple instances of CIM\_HardwareThread, implementations shall utilize  
389 the following algorithm to produce the natural (that is, default) ordering:

- 390 • Results for CIM\_HardwareThread are unordered; therefore, no algorithm is defined.

### 391 6.3.2 Reset

392 This section describes how to implement the `reset` verb when applied to an instance of  
393 CIM\_HardwareThread. Implementations may support the use of the `reset` verb with  
394 CIM\_HardwareThread.

#### 395 6.3.2.1 Command Form

```
396 reset <CIM_HardwareThread single instance>
```

##### 397 6.3.2.1.1 CIM Requirements

```
398 uint16 EnabledState;
399 uint16 RequestedState;
400 uint32 CIM_HardwareThread.RequestStateChange (
401     [IN] uint16 RequestedState,
402     [OUT] REF CIM_ConcreteJob Job,
403     [IN] datetime TimeoutPeriod );
```

##### 404 6.3.2.1.2 Behavior Requirements

###### 405 6.3.2.1.2.1 Preconditions

406 In this section `$instance` represents the targeted instance of CIM\_HardwareThread.

```
407 $instance=<CIM_HardwareThread single instance>;
```

###### 408 6.3.2.1.2.2 Pseudo Code

```
409 &smResetRSC ( $instance.getObjectPath() );
410 &smEnd;
```

### 411 6.3.3 Set

412 This section describes how to implement the `set` verb when it is applied to an instance of  
413 CIM\_HardwareThread. Implementations may support the use of the `set` verb with CIM\_HardwareThread.

414 The `set` verb is used to modify descriptive properties of the CIM\_HardwareThread instance.

#### 415 6.3.3.1 General Usage of Set for a Single Property

416 This command form corresponds to the general usage of the `set` verb to modify a single property of a  
417 target instance. This is the most common case.

418 The requirement for supporting modification of a property using this command form shall be equivalent to  
419 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
420 in the [CPU Profile](#).

##### 421 6.3.3.1.1 Command Form

```
422 set <CIM_HardwareThread single instance> <propertyname>=<propertyvalue>
```



### 423 6.3.3.1.2 CIM Requirements

424 See CIM\_HardwareThread in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
425 properties.

### 426 6.3.3.1.3 Behavior Requirements

```
427 $instance=<CIM_HardwareThread single instance>
428 #propertyNames[] = {<propertyname>};
429 #propertyValues[] = {<propertyvalue>};
430 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
431 &smEnd;
```

### 432 6.3.3.2 General Usage of Set for Multiple Properties

433 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
434 target instance where there is not an explicit relationship between the properties. This is the most  
435 common case.

436 The requirement for supporting modification of a property using this command form shall be equivalent to  
437 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
438 in the [CPU Profile](#).

#### 439 6.3.3.2.1 Command Form

```
440 set <CIM_HardwareThread single instance> <propertyname1>=<propertyvalue1>
441 <propertynamen>=<propertyvaluen>
```

#### 442 6.3.3.2.2 CIM Requirements

443 See CIM\_HardwareThread in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
444 properties.

#### 445 6.3.3.2.3 Behavior Requirements

```
446 $instance=<CIM_HardwareThread single instance>
447 #propertyNames[] = {<propertyname>};
448 for #i < n
449 {
450     #propertyNames[#i] = <propertyname#i>
451     #propertyValues[#i] = <propertyvalue#i>
452 }
453 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
454 &smEnd;
```

### 455 6.3.4 Show

456 This section describes how to implement the `show` verb when applied to an instance of  
457 CIM\_HardwareThread. Implementations shall support the use of the `show` verb with  
458 CIM\_HardwareThread.

#### 459 6.3.4.1 Show Command Form for Multiple Instances Target

460 This command form is used to show many instances of CIM\_HardwareThread.

##### 461 6.3.4.1.1 Command Form

```
462 show <CIM_HardwareThread multiple instances>
```

463 **6.3.4.1.2 CIM Requirements**

464 See CIM\_HardwareThread in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
465 properties.

466 **6.3.4.1.3 Behavior Requirements**467 **6.3.4.1.3.1 Preconditions**

468 In this section \$containerInstance represents the instance of CIM\_ProcessorCore which represents  
469 the container system and is associated to the targeted instances of CIM\_HardwareThread through the  
470 CIM\_ConcreteComponent association.

471 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

472 **6.3.4.1.3.2 Pseudo Code**

```
473 #propertylist[] = NULL;
474 if ( false == #all)
475 {
476     #propertylist[] = <array of mandatory non-key property names (see CIM
477         Requirements)>;
478 }
479 &smShowInstances ( "CIM_HardwareThread", "CIM_ConcreteComponent" , NULL, NULL,
480     $containerInstance.getObjectPath(), NULL, NULL, #propertylist[] );
481 &smEnd;
```

482 **6.3.4.2 Show Command Form for a Single Instance Target**

483 This command form is used to show a single instance of CIM\_HardwareThread.

484 **6.3.4.2.1 Command Form**

```
485 show <CIM_HardwareThread single instance>
```

486 **6.3.4.2.2 CIM Requirements**

487 See CIM\_HardwareThread in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
488 properties.

489 **6.3.4.2.3 Behavior Requirements**490 **6.3.4.2.3.1 Preconditions**

491 In this section \$instance represents the targeted instance of CIM\_HardwareThread.

```
492 $instance=<CIM_HardwareThread single instance>;
```

493 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

494 **6.3.4.2.3.2 Pseudo Code**

```
495 #propertylist[] = NULL;
496 if ( false == #all)
497 {
498     #propertylist[] = <array of mandatory non-key property names (see CIM
499         Requirements)>;
500 }
501 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
502 &smEnd;
```

### 503 6.3.5 Start

504 This section describes how to implement the `start` verb when applied to an instance of  
505 `CIM_HardwareThread`. Implementations may support the use of the `start` verb with  
506 `CIM_HardwareThread`.

#### 507 6.3.5.1.1 Command Form

```
508 start <CIM_HardwareThread single instance>
```

#### 509 6.3.5.1.2 CIM Requirements

```
510 uint16 EnabledState;
511 uint16 RequestedState;
512 uint32 CIM_HardwareThread.RequestStateChange (
513     [IN] uint16 RequestedState,
514     [OUT] REF CIM_ConcreteJob Job,
515     [IN] datetime TimeoutPeriod );
```

##### 516 6.3.5.1.2.1 Behavior Requirements

##### 517 6.3.5.1.2.2 Preconditions

518 In this section `$instance` represents the targeted instance of `CIM_HardwareThread`.

```
519 $instance=<CIM_HardwareThread single instance>;
```

##### 520 6.3.5.1.2.3 Pseudo Code

```
521 &smStartRSC ( $instance.getObjectPath() );
522 &smEnd;
```

### 523 6.3.6 Stop

524 This section describes how to implement the `stop` verb when applied to an instance of  
525 `CIM_HardwareThread`. Implementations may support the use of the `stop` verb with  
526 `CIM_HardwareThread`.

#### 527 6.3.6.1.1 Command Form

```
528 stop <CIM_HardwareThread single instance>
```

#### 529 6.3.6.1.2 CIM Requirements

```
530 uint16 EnabledState;
531 uint16 RequestedState;
532 uint32 CIM_HardwareThread.RequestStateChange (
533     [IN] uint16 RequestedState,
534     [OUT] REF CIM_ConcreteJob Job,
535     [IN] datetime TimeoutPeriod );
```

##### 536 6.3.6.1.3 Behavior Requirements

##### 537 6.3.6.1.3.1 Preconditions

538 In this section `$instance` represents the targeted instance of `CIM_HardwareThread`.

```
539 $instance=<CIM_HardwareThread single instance>;
```

540 **6.3.6.1.3.2 Pseudo Code**

```
541 &smStopRSC ( $instance.getObjectPath() );
542 &smEnd;
```

543 **6.4 CIM\_Memory**

544 The CLP implementation shall expose only a unique path for each instance of CIM\_Memory. The  
545 association class for the path shall be CIM\_AssociatedCacheMemory.

546 The `cd`, `exit`, `help` and `version` verbs shall be supported as described in [DSP0216](#).

547 Table 4 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
548 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and  
549 target. Table 4 is for informational purposes only; in case of a conflict between Table 4 and requirements  
550 detailed in the following sections, the text detailed in the following sections supersedes the information in  
551 Table 4.

552 **Table 4 – Command Verb Requirements for CIM\_Memory**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.4.2.
Set	May	See 6.4.3.
Show	Shall	See 6.4.4.
Start	May	See 6.4.5.
Stop	May	See 6.4.6.

553 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, and `load`.

554 **6.4.1 Ordering of Results**

555 When results are returned for multiple instances of CIM\_Memory, implementations shall utilize the  
556 following algorithm to produce the natural (that is, default) ordering:

- 557 • Results for CIM\_Memory are unordered; therefore, no algorithm is defined.

558 **6.4.2 Reset**

559 This section describes how to implement the `reset` verb when applied to an instance of CIM\_Memory.  
560 Implementations may support the use of the `reset` verb with CIM\_Memory.

561 **6.4.2.1.1 Command Form**

```
562 reset <CIM_Memory single instance>
```

### 563 6.4.2.1.2 CIM Requirements

```
564 uint16 EnabledState;
565 uint16 RequestedState;
566 uint32 CIM_Memory.RequestStateChange (
567     [IN] uint16 RequestedState,
568     [OUT] REF CIM_ConcreteJob Job,
569     [IN] datetime TimeoutPeriod );
```

### 570 6.4.2.1.3 Behavior Requirements

#### 571 6.4.2.1.3.1 Preconditions

572 In this section `$instance` represents the targeted instance of `CIM_Memory`.

```
573 $instance=<CIM_Memory single instance>;
```

#### 574 6.4.2.1.3.2 Pseudo Code

```
575 &smResetRSC ( $instance.getObjectPath() );
576 &smEnd;
```

## 577 6.4.3 Set

578 This section describes how to implement the `set` verb when it is applied to an instance of `CIM_Memory`.  
579 Implementations may support the use of the `set` verb with `CIM_Memory`.

580 The `set` verb is used to modify descriptive properties of the `CIM_Memory` instance.

### 581 6.4.3.1 General Usage of Set for a Single Property

582 This command form corresponds to the general usage of the `set` verb to modify a single property of a  
583 target instance. This is the most common case.

584 The requirement for supporting modification of a property using this command form shall be equivalent to  
585 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
586 in the [CPU Profile](#).

#### 587 6.4.3.1.1 Command Form

```
588 set <CIM_Memory single instance> <propertyname>=<propertyvalue>
```

#### 589 6.4.3.1.2 CIM Requirements

590 See `CIM_Memory` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.

#### 591 6.4.3.1.3 Behavior Requirements

```
592 $instance=<CIM_Memory single instance>
593 #propertyName[] = {<propertyname>};
594 #propertyValues[] = {<propertyvalue>};
595 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
596 &smEnd;
```

### 597 6.4.3.2 General Usage of Set for Multiple Properties

598 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
599 target instance where there is not an explicit relationship between the properties. This is the most  
600 common case.

601 The requirement for supporting modification of a property using this command form shall be equivalent to  
 602 the requirement for supporting modification of the property using the ModifyInstance operation as defined  
 603 in the [CPU Profile](#).

#### 604 6.4.3.2.1 Command Form

```
605 set <CIM_Memory single instance> <propertyname1>=<propertyvalue1>
606 <propertynamen>=<propertyvaluen>
```

#### 607 6.4.3.2.2 CIM Requirements

608 See CIM\_Memory in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.

#### 609 6.4.3.2.3 Behavior Requirements

```
610 $instance=<CIM_Memory single instance>
611 #propertyName[] = {<propertyname>};
612 for #i < n
613 {
614     #propertyName[#i] = <propertyname#i>
615     #propertyValue[#i] = <propertyvalue#i>
616 }
617 &smSetInstance ( $instance, #propertyName[], #propertyValue[] );
618 &smEnd;
```

### 619 6.4.4 Show

620 This section describes how to implement the `show` verb when applied to an instance of CIM\_Memory.  
 621 Implementations shall support the use of the `show` verb with CIM\_Memory.

#### 622 6.4.4.1 Show Command Form for Multiple Instances Target

623 This command form is used to show many instances of CIM\_Memory.

##### 624 6.4.4.1.1 Command Form

```
625 show <CIM_Memory multiple instances>
```

##### 626 6.4.4.1.2 CIM Requirements

627 See CIM\_Memory in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.

##### 628 6.4.4.1.3 Behavior Requirements

###### 629 6.4.4.1.3.1 Preconditions

630 In this section `$containerInstance` represents the instance of CIM\_Processor or CIM\_ProcessorCore  
 631 which represents the processor or the core that utilizes the cache memory and is associated to the  
 632 targeted instances of CIM\_Memory through the CIM\_AssociatedCacheMemory association.

633 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

###### 634 6.4.4.1.3.2 Pseudo Code

```
635 #propertylist[] = NULL;
636 if ( false == #all)
637 {
638     #propertylist[] = <array of mandatory non-key property names (see CIM
639     Requirements)>;
640 }
```

```

641 &smShowInstances ( "CIM_Memory", "CIM_AssociatedCacheMemory" , NULL, NULL,
642     $containerInstance.getObjectPath(), NULL, NULL, #propertylist[] );
643 &smEnd;

```

#### 644 **6.4.4.2 Show Command Form for a Single Instance Target**

645 This command form is used to show a single instance of CIM\_Memory.

##### 646 **6.4.4.2.1 Command Form**

```
647 show <CIM_Memory single instance>
```

##### 648 **6.4.4.2.2 CIM Requirements**

649 See CIM\_Memory in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.

##### 650 **6.4.4.2.3 Behavior Requirements**

###### 651 **6.4.4.2.3.1 Preconditions**

652 In this section \$instance represents the targeted instance of CIM\_Memory.

```
653 $instance=<CIM_Memory single instance>;
```

654 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

###### 655 **6.4.4.2.3.2 Pseudo Code**

```

656 #propertylist[] = NULL;
657 if ( false == #all)
658 {
659     #propertylist[] = <array of mandatory non-key property names (see CIM
660         Requirements)>;
661 }
662 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
663 &smEnd;

```

#### 664 **6.4.5 Start**

665 This section describes how to implement the `start` verb when applied to an instance of CIM\_Memory.  
666 Implementations may support the use of the `start` verb with CIM\_Memory.

##### 667 **6.4.5.1.1 Command Form**

```
668 start <CIM_Memory single instance>
```

##### 669 **6.4.5.1.2 CIM Requirements**

```

670 uint16 EnabledState;
671 uint16 RequestedState;
672 uint32 CIM_Memory.RequestStateChange (
673     [IN] uint16 RequestedState,
674     [OUT] REF CIM_ConcreteJob Job,
675     [IN] datetime TimeoutPeriod );

```

##### 676 **6.4.5.1.3 Behavior Requirements**

###### 677 **6.4.5.1.3.1 Preconditions**

678 In this section \$instance represents the targeted instance of CIM\_Memory.

```
679 $instance=<CIM_Memory single instance>;
```

680 **6.4.5.1.3.2 Pseudo Code**

```
681 &smStartRSC ( $instance.GetObjectPath() );
682 &smEnd;
```

683 **6.4.6 Stop**

684 This section describes how to implement the `stop` verb when applied to an instance of `CIM_Memory`.  
 685 Implementations may support the use of the `stop` verb with `CIM_Memory`.

686 **6.4.7 Command Form**

```
687 stop <CIM_Memory single instance>
```

688 **6.4.7.1.1 CIM Requirements**

```
689 uint16 EnabledState;
690 uint16 RequestedState;
691 uint32 CIM_Memory.RequestStateChange (
692     [IN] uint16 RequestedState,
693     [OUT] REF CIM_ConcreteJob Job,
694     [IN] datetime TimeoutPeriod );
```

695 **6.4.7.1.2 Behavior Requirements**696 **6.4.7.1.2.1 Preconditions**

697 In this section `$instance` represents the targeted instance of `CIM_Memory`.

```
698 $instance=<CIM_Memory single instance>;
```

699 **6.4.7.1.2.2 Pseudo Code**

```
700 &smStopRSC ( $instance.GetObjectPath() );
701 &smEnd;
```

702 **6.5 CIM\_ElementCapabilities**

703 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

704 Table 5 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 705 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 706 target. Table 5 is for informational purposes only; in case of a conflict between Table 5 and requirements  
 707 detailed in the following sections, the text detailed in the following sections supersedes the information in  
 708 Table 5.

709 **Table 5 – Command Verb Requirements for CIM\_ElementCapabilities**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.5.2.



Command Verb	Requirement	Comments
Start	Not supported	
Stop	Not supported	

710 No mapping is defined for the following verbs for the specified target: create, delete, dump, exit,  
711 load, reset, set, start, and stop.

### 712 6.5.1 Ordering of Results

713 When results are returned for multiple instances of CIM\_ElementCapabilities, implementations shall  
714 utilize the following algorithm to produce the natural (that is, default) ordering:

- 715 • Results for CIM\_ElementCapabilities are unordered; therefore, no algorithm is defined.

### 716 6.5.2 Show

717 This section describes how to implement the `show` verb when applied to an instance of  
718 CIM\_ElementCapabilities. Implementations shall support the use of the `show` verb with  
719 CIM\_ElementCapabilities.

#### 720 6.5.2.1 Show Command Form for Multiple Instances Target – 721 CIM\_EnabledLogicalElementCapabilities Reference

722 This command form is used to show many instances of CIM\_ElementCapabilities. This command form  
723 corresponds to a `show` command issued against instances of CIM\_ElementCapabilities where only one  
724 reference is specified and the reference is to an instance of CIM\_EnabledLogicalElementCapabilities.

##### 725 6.5.2.1.1 Command Form

```
726 show <CIM_ElementCapabilities multiple instances>
```

##### 727 6.5.2.1.2 CIM Requirements

728 See CIM\_ElementCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
729 properties.

##### 730 6.5.2.1.3 Behavior Requirements

###### 731 6.5.2.1.3.1 Preconditions

732 In this section `$instance` represents the instance of CIM\_EnabledLogicalElementCapabilities which is  
733 referenced by CIM\_ElementCapabilities.

###### 734 6.5.2.1.3.2 Pseudo Code

```
735 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
736 &smEnd;
```

#### 737 6.5.2.2 Show Command Form for Multiple Instances Target – CIM\_ProcessorCapabilities 738 Reference

739 This command form is used to show many instances of CIM\_ElementCapabilities. This command form  
740 corresponds to a `show` command issued against instances of CIM\_ElementCapabilities where only one  
741 reference is specified and the reference is to an instance of CIM\_ProcessorCapabilities.

##### 742 6.5.2.2.1 Command Form

```
743 show <CIM_ElementCapabilities multiple instances>
```

#### 744 6.5.2.2.2 CIM Requirements

745 See CIM\_ElementCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
746 properties.

#### 747 6.5.2.2.3 Behavior Requirements

##### 748 6.5.2.2.3.1 Preconditions

749 In this section `$instance` represents the instance of CIM\_ProcessorCapabilities which is referenced by  
750 CIM\_ElementCapabilities.

##### 751 6.5.2.2.3.2 Pseudo Code

```
752 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
753 &smEnd;
```

#### 754 6.5.2.3 Show Command Form for a Single Instance – CIM\_Processor Reference

755 This command form is used to show a single instance of CIM\_ElementCapabilities. This command form  
756 corresponds to a `show` command issued against a single instance of CIM\_ElementCapabilities where  
757 only one reference is specified and the reference is to the instance of CIM\_Processor.

##### 758 6.5.2.3.1 Command Form

```
759 show <CIM_ElementCapabilities single instance>
```

##### 760 6.5.2.3.2 CIM Requirements

761 See CIM\_ElementCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
762 properties.

##### 763 6.5.2.3.3 Behavior Requirements

##### 764 6.5.2.3.3.1 Preconditions

765 In this section `$instance` represents the instance of CIM\_Processor which is referenced by  
766 CIM\_ElementCapabilities.

##### 767 6.5.2.3.3.2 Pseudo Code

```
768 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
769 &smEnd;
```

#### 770 6.5.2.4 Show Command Form for a Single Instance – CIM\_ProcessorCore Reference

771 This command form is used to show a single instance of CIM\_ElementCapabilities. This command form  
772 corresponds to a `show` command issued against a single instance of CIM\_ElementCapabilities where  
773 only one reference is specified and the reference is to the instance of CIM\_ProcessorCore.

##### 774 6.5.2.4.1 Command Form

```
775 show <CIM_ElementCapabilities single instance>
```

##### 776 6.5.2.4.2 CIM Requirements

777 See CIM\_ElementCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
778 properties.

### 779 6.5.2.4.3 Behavior Requirements

#### 780 6.5.2.4.3.1 Preconditions

781 In this section `$instance` represents the instance of `CIM_ProcessorCore` which is referenced by  
782 `CIM_ElementCapabilities`.

#### 783 6.5.2.4.3.2 Pseudo Code

```
784 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.GetObjectPath() );  
785 &smEnd;
```

### 786 6.5.2.5 Show Command Form for a Single Instance – CIM\_HardwareThread Reference

787 This command form is used to show a single instance of `CIM_ElementCapabilities`. This command form  
788 corresponds to a `show` command issued against a single instance of `CIM_ElementCapabilities` where  
789 only one reference is specified and the reference is to the instance of `CIM_HardwareThread`.

#### 790 6.5.2.5.1 Command Form

```
791 show <CIM_ElementCapabilities single instance>
```

#### 792 6.5.2.5.2 CIM Requirements

793 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
794 properties.

### 795 6.5.2.5.3 Behavior Requirements

#### 796 6.5.2.5.3.1 Preconditions

797 In this section `$instance` represents the instance of `CIM_HardwareThread` which is referenced by  
798 `CIM_ElementCapabilities`.

#### 799 6.5.2.5.3.2 Pseudo Code

```
800 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.GetObjectPath() );  
801 &smEnd;
```

### 802 6.5.2.6 Show Command Form for a Single Instance – CIM\_Memory Reference

803 This command form is used to show a single instance of `CIM_ElementCapabilities`. This command form  
804 corresponds to a `show` command issued against a single instance of `CIM_ElementCapabilities` where  
805 only one reference is specified and the reference is to the instance of `CIM_Memory`.

#### 806 6.5.2.6.1 Command Form

```
807 show <CIM_ElementCapabilities single instance>
```

#### 808 6.5.2.6.2 CIM Requirements

809 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
810 properties.

### 811 6.5.2.6.3 Behavior Requirements

#### 812 6.5.2.6.3.1 Preconditions

813 In this section `$instance` represents the instance of `CIM_Memory` which is referenced by  
814 `CIM_ElementCapabilities`.

**815 6.5.2.6.3.2 Pseudo Code**

```
816 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
817 &smEnd;
```

**818 6.5.2.7 Show Command Form for a Single Instance Target – Both References:  
819 CIM\_ProcessorCapabilities and CIM\_Processor**

820 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
821 the `show` command issued against `CIM_ElementCapabilities` where both references are specified and  
822 therefore the desired instance is unambiguously identified.

**823 6.5.2.7.1 Command Form**

```
824 show <CIM_ElementCapabilities single instance>
```

**825 6.5.2.7.2 CIM Requirements**

826 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
827 properties.

**828 6.5.2.7.3 Behavior Requirements****829 6.5.2.7.3.1 Preconditions**

830 In this section `$instanceA` represents the referenced instance of `CIM_Processor` through  
831 `CIM_ElementCapabilities` association. `$instanceB` represents the instance of  
832 `CIM_ProcessorCapabilities` which is referenced by `CIM_ElementCapabilities`.

**833 6.5.2.7.3.2 Pseudo Code**

```
834 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),  
835     $instanceB.getObjectPath() );  
836 &smEnd;
```

**837 6.5.2.8 Show Command Form for a Single Instance Target – Both References:  
838 CIM\_EnabledLogicalElementCapabilities and CIM\_ProcessorCore**

839 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
840 the `show` command issued against `CIM_ElementCapabilities` where both references are specified and  
841 therefore the desired instance is unambiguously identified.

**842 6.5.2.8.1 Command Form**

```
843 show <CIM_ElementCapabilities single instance>
```

**844 6.5.2.8.2 CIM Requirements**

845 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
846 properties.

**847 6.5.2.8.3 Behavior Requirements****848 6.5.2.8.3.1 Preconditions**

849 In this section `$instanceA` represents the referenced instance of `CIM_ProcessorCore` through  
850 `CIM_ElementCapabilities` association. `$instanceB` represents the instance of  
851 `CIM_EnabledLogicalElementCapabilities` which is referenced by `CIM_ElementCapabilities`.

### 852 **6.5.2.8.3.2 Pseudo Code**

```
853 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
854     $instanceB.getObjectPath() );
855 &smEnd;
```

### 856 **6.5.2.9 Show Command Form for a Single Instance Target – Both References: 857 CIM\_EnabledLogicalElementCapabilities and CIM\_HardwareThread**

858 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
859 the `show` command issued against `CIM_ElementCapabilities` where both references are specified and  
860 therefore the desired instance is unambiguously identified.

#### 861 **6.5.2.9.1 Command Form**

```
862 show <CIM_ElementCapabilities single instance>
```

#### 863 **6.5.2.9.2 CIM Requirements**

864 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
865 properties.

#### 866 **6.5.2.9.3 Behavior Requirements**

##### 867 **6.5.2.9.3.1 Preconditions**

868 In this section `$instanceA` represents the referenced instance of `CIM_HardwareThread` through  
869 `CIM_ElementCapabilities` association. `$instanceB` represents the instance of  
870 `CIM_EnabledLogicalElementCapabilities` which is referenced by `CIM_ElementCapabilities`.

##### 871 **6.5.2.9.3.2 Pseudo Code**

```
872 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
873     $instanceB.getObjectPath() );
874 &smEnd;
```

### 875 **6.5.2.10 Show Command Form for a Single Instance Target – Both References: 876 CIM\_EnabledLogicalElementCapabilities and CIM\_Memory**

877 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
878 the `show` command issued against `CIM_ElementCapabilities` where both references are specified and  
879 therefore the desired instance is unambiguously identified.

#### 880 **6.5.2.10.1 Command Form**

```
881 show <CIM_ElementCapabilities single instance>
```

#### 882 **6.5.2.10.2 CIM Requirements**

883 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
884 properties.

#### 885 **6.5.2.10.3 Behavior Requirements**

##### 886 **6.5.2.10.3.1 Preconditions**

887 In this section `$instanceA` represents the referenced instance of `CIM_Memory` through  
888 `CIM_ElementCapabilities` association. `$instanceB` represents the instance of  
889 `CIM_EnabledLogicalElementCapabilities` which is referenced by `CIM_ElementCapabilities`.

890 **6.5.2.10.3.2 Pseudo Code**

```
891 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
892     $instanceB.getObjectPath() );
893 &smEnd;
```

894 **6.6 CIM\_EnabledLogicalElementCapabilities**

895 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

896 Table 6 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 897 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 898 target. Table 6 is for informational purposes only; in case of a conflict between Table 6 and requirements  
 899 detailed in the following sections, the text detailed in the following sections supersedes the information in  
 900 Table 6.

901 **Table 6 – Command Verb Requirements for CIM\_EnabledLogicalElementCapabilities**

902

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.6.2.
Start	Not supported	
Stop	Not supported	

903 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 904 `reset`, `start`, and `stop`.

905 **6.6.1 Ordering of Results**

906 When results are returned for multiple instances of `CIM_EnabledLogicalElementCapabilities`,  
 907 implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

- 908 • Results for `CIM_EnabledLogicalElementCapabilities` are unordered: therefore, no algorithm is  
 909 defined.

910 **6.6.2 Show**

911 This section describes how to implement the `show` verb when applied to an instance of  
 912 `CIM_EnabledLogicalElementCapabilities`. Implementations shall support the use of the `show` verb with  
 913 `CIM_EnabledLogicalElementCapabilities`.

914 **6.6.2.1 Show Command Form for Multiple Instances Target**

915 This command form is used to show many instances of `CIM_EnabledLogicalElementCapabilities`.

916 **6.6.2.1.1 Command Form**

```
917 show <CIM_EnabledLogicalElementCapabilities multiple instances>
```

### 918 6.6.2.1.2 CIM Requirements

919 See CIM\_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list  
920 of mandatory properties.

### 921 6.6.2.1.3 Behavior Requirements

#### 922 6.6.2.1.3.1 Preconditions

923 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection with  
924 ElementName property that contains “Capabilities” and is associated to the targeted instances of  
925 CIM\_EnabledLogicalElementCapabilities through the CIM\_MemberOfCollection association.

926 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

#### 927 6.6.2.1.3.2 Pseudo Code

```
928 #propertylist[] = NULL;
929 if ( false == #all)
930 {
931     #propertylist[] = <array of mandatory non-key property names (see CIM
932         Requirements)>;
933 }
934 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
935     NULL, NULL, $containerInstance.getObjectPath(), NULL, NULL, #propertylist[] );
936 &smEnd;
```

### 937 6.6.2.2 Show Command Form for a Single Instance Target

938 This command form is used to show a single instance of CIM\_EnabledLogicalElementCapabilities.

#### 939 6.6.2.2.1 Command Form

```
940 show <CIM_EnabledLogicalElementCapabilities single instance>
```

#### 941 6.6.2.2.2 CIM Requirements

942 See CIM\_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list  
943 of mandatory properties.

#### 944 6.6.2.2.3 Behavior Requirements

##### 945 6.6.2.2.3.1 Preconditions

946 In this section \$instance represents the targeted instance of CIM\_EnabledLogicalElementCapabilities.

```
947 $instance=<CIM_EnabledLogicalElementCapabilities single instance>;
```

948 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

##### 949 6.6.2.2.3.2 Pseudo Code

```
950 #propertylist[] = NULL;
951 if ( false == #all)
952 {
953     #propertylist[] = <array of mandatory non-key property names (see CIM
954         Requirements)>;
955 }
956 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
957 &smEnd;
```

958 **6.7 CIM\_Processor**959 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

960 Table 7 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 961 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 962 target. Table 7 is for informational purposes only; in case of a conflict between Table 7 and requirements  
 963 detailed in the following sections, the text detailed in the following sections supersedes the information in  
 964 Table 7.

965 **Table 7 – Command Verb Requirements for CIM\_Processor**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.7.2.
Set	May	See 6.7.3.
Show	Shall	See 6.7.4.
Start	May	See 6.7.5.
Stop	May	See 6.7.6.

966 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, and `load`.967 **6.7.1 Ordering of Results**968 When results are returned for multiple instances of `CIM_Processor`, implementations shall utilize the  
 969 following algorithm to produce the natural (that is, default) ordering:

- 970
- Results for `CIM_Processor` are unordered; therefore, no algorithm is defined.

971 **6.7.2 Reset**972 This section describes how to implement the `reset` verb when applied to an instance of `CIM_Processor`.  
 973 Implementations may support the use of the `reset` verb with `CIM_Processor`.974 **6.7.2.1 Command Form**975 `reset <CIM_Processor single instance>`976 **6.7.2.2 CIM Requirements**

```

977 uint16 EnabledState;
978 uint16 RequestedState;
979 uint32 CIM_Processor.RequestStateChange (
980     [IN] uint16 RequestedState,
981     [OUT] REF CIM_ConcreteJob Job,
982     [IN] datetime TimeoutPeriod );

```



### 983 6.7.2.3 Behavior Requirements

#### 984 6.7.2.3.1 Preconditions

985 In this section `$instance` represents the targeted instance of `CIM_Processor`.

```
986 $instance=<CIM_Processor single instance>;
```

#### 987 6.7.2.3.2 Pseudo Code

```
988 &smResetRSC ( $instance.GetObjectPath() );
989 &smEnd;
```

### 990 6.7.3 Set

991 This section describes how to implement the `set` verb when it is applied to an instance of  
992 `CIM_Processor`. Implementations may support the use of the `set` verb with `CIM_Processor`.

993 The `set` verb is used to modify descriptive properties of the `CIM_Processor` instance.

#### 994 6.7.3.1 General Usage of Set for a Single Property

995 This command form corresponds to the general usage of the `set` verb to modify a single property of a  
996 target instance. This is the most common case.

997 The requirement for supporting modification of a property using this command form shall be equivalent to  
998 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
999 in the [CPU Profile](#).

#### 1000 6.7.3.2 Command Form

```
1001 set <CIM_Processor single instance> <propertyname>=<propertyvalue>
```

#### 1002 6.7.3.3 CIM Requirements

1003 See `CIM_Processor` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.

#### 1004 6.7.3.4 Behavior Requirements

```
1005 $instance=<CIM_Processor single instance>
1006 #propertyName[] = {<propertyname>};
1007 #propertyValues[] = {<propertyvalue>};
1008 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1009 &smEnd;
```

#### 1010 6.7.3.5 General Usage of Set for Multiple Properties

1011 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
1012 target instance where there is not an explicit relationship between the properties. This is the most  
1013 common case.

1014 The requirement for supporting modification of a property using this command form shall be equivalent to  
1015 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
1016 in the [CPU Profile](#).

##### 1017 6.7.3.5.1 Command Form

```
1018 set <CIM_Processor single instance> <propertyname1>=<propertyvalue1>
1019 <propertynamen>=<propertyvaluen>
```

1020 **6.7.3.5.2 CIM Requirements**1021 See CIM\_Processor in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.1022 **6.7.3.5.3 Behavior Requirements**

```

1023 $instance=<CIM_Processor single instance>
1024 #propertyName[] = {<propertyname>};
1025 for #i < n
1026 {
1027     #propertyName[#i] = <propertyname#i>
1028     #propertyValue[#i] = <propertyvalue#i>
1029 }
1030 &smSetInstance ( $instance, #propertyName[], #propertyValue[] );
1031 &smEnd;

```

1032 **6.7.4 Show**

1033 This section describes how to implement the `show` verb when applied to an instance of CIM\_Processor.  
 1034 Implementations shall support the use of the `show` verb with CIM\_Processor.

1035 **6.7.4.1 Show Command Form for Multiple Instances Target**

1036 This command form is used to show many instances of CIM\_Processor.

1037 **6.7.4.1.1 Command Form**1038 `show <CIM_Processor multiple instances>`1039 **6.7.4.1.2 CIM Requirements**1040 See CIM\_Processor in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.1041 **6.7.4.1.3 Behavior Requirements**1042 **6.7.4.1.3.1 Preconditions**

1043 In this section `$containerInstance` represents the instance of CIM\_ComputerSystem which  
 1044 represents the container system and is associated to the targeted instances of CIM\_Processor through  
 1045 the CIM\_SystemDevice association.

1046 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.1047 **6.7.4.1.3.2 Pseudo Code**

```

1048 #propertylist[] = NULL;
1049 if ( false == #all)
1050 {
1051     #propertylist[] = <array of mandatory non-key property names (see CIM
1052     Requirements)>;
1053 }
1054 &smShowInstances ( "CIM_Processor", "CIM_SystemDevice", NULL, NULL,
1055     $containerInstance.getObjectPath(), NULL, NULL, #propertylist[] );
1056 &smEnd;

```

1057 **6.7.4.2 Show Command Form for a Single Instance Target**

1058 This command form is used to show a single instance of CIM\_Processor.

1059 **6.7.4.2.1 Command Form**1060 `show <CIM_Processor single instance>`1061 **6.7.4.2.2 CIM Requirements**1062 See CIM\_Processor in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory properties.1063 **6.7.4.2.3 Behavior Requirements**1064 **6.7.4.2.3.1 Preconditions**1065 In this section `$instance` represents the targeted instance of CIM\_Processor.1066 `$instance=<CIM_Processor single instance>;`1067 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.1068 **6.7.4.2.3.2 Pseudo Code**

```

1069 #propertylist[] = NULL;
1070 if ( false == #all)
1071 {
1072     #propertylist[] = <array of mandatory non-key property names (see CIM
1073     Requirements)>;
1074 }
1075 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1076 &smEnd;
```

1077 **6.7.5 Start**

1078 This section describes how to implement the `start` verb when applied to an instance of CIM\_Processor.  
 1079 Implementations may support the use of the `start` verb with CIM\_Processor.

1080 **6.7.5.1 Command Form**1081 `start <CIM_Processor single instance>`1082 **6.7.5.2 CIM Requirements**

```

1083 uint16 EnabledState;
1084 uint16 RequestedState;
1085 uint32 CIM_Processor.RequestStateChange (
1086     [IN] uint16 RequestedState,
1087     [OUT] REF CIM_ConcreteJob Job,
1088     [IN] datetime TimeoutPeriod );
```

1089 **6.7.5.3 Behavior Requirements**1090 **6.7.5.3.1 Preconditions**1091 In this section `$instance` represents the targeted instance of CIM\_Processor.1092 `$instance=<CIM_Processor single instance>;`1093 **6.7.5.3.2 Pseudo Code**

```

1094 &smStartRSC ( $instance.getObjectPath() );
1095 &smEnd;
```

1096 **6.7.6 Stop**

1097 This section describes how to implement the `stop` verb when applied to an instance of `CIM_Processor`.  
 1098 Implementations may support the use of the `stop` verb with `CIM_Processor`.

1099 **6.7.6.1 Command Form**

```
1100 stop <CIM_Processor single instance>
```

1101 **6.7.6.2 CIM Requirements**

```
1102 uint16 EnabledState;
1103 uint16 RequestedState;
1104 uint32 CIM_Processor.RequestStateChange (
1105     [IN] uint16 RequestedState,
1106     [OUT] REF CIM_ConcreteJob Job,
1107     [IN] datetime TimeoutPeriod );
```

1108 **6.7.6.3 Behavior Requirements**1109 **6.7.6.3.1 Preconditions**

1110 In this section `$instance` represents the targeted instance of `CIM_Processor`.

```
1111 $instance=<CIM_Processor single instance>;
```

1112 **6.7.6.3.2 Pseudo Code**

```
1113 &smStopRSC ( $instance.getObjectPath() );
1114 &smEnd;
```

1115 **6.8 CIM\_ProcessorCapabilities**

1116 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1117 Table 8 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1118 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1119 target. Table 8 is for informational purposes only; in case of a conflict between Table 8 and requirements  
 1120 detailed in the following sections, the text detailed in the following sections supersedes the information in  
 1121 Table 8.

1122 **Table 8 – Command Verb Requirements for CIM\_ProcessorCapabilities**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.8.2.
Start	Not supported	
Stop	Not supported	

1123 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 1124 `reset`, `start`, and `stop`.

## 1125 6.8.1 Ordering of Results

1126 When results are returned for multiple instances of CIM\_ProcessorCapabilities, implementations shall  
1127 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1128 • Results for CIM\_ProcessorCapabilities are unordered; therefore, no algorithm is defined.

## 1129 6.8.2 Show

1130 This section describes how to implement the `show` verb when applied to an instance of  
1131 CIM\_ProcessorCapabilities. Implementations shall support the use of the `show` verb with  
1132 CIM\_ProcessorCapabilities.

### 1133 6.8.2.1 Show Command Form for Multiple Instances Target

1134 This command form is used to show many instances of CIM\_ProcessorCapabilities.

#### 1135 6.8.2.1.1 Command Form

```
1136 show <CIM_ProcessorCapabilities multiple instances>
```

#### 1137 6.8.2.1.2 CIM Requirements

1138 See CIM\_ProcessorCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
1139 properties.

#### 1140 6.8.2.1.3 Behavior Requirements

##### 1141 6.8.2.1.3.1 Preconditions

1142 In this section `$containerInstance` represents the instance of CIM\_ConcreteCollection with  
1143 ElementName property that contains “Capabilities” and is associated to the targeted instances of  
1144 CIM\_ProcessorCapabilities through the CIM\_MemberOfCollection association.

1145 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

##### 1146 6.8.2.1.3.2 Pseudo Code

```
1147 #propertylist[] = NULL;
1148 if ( false == #all)
1149 {
1150     #propertylist[] = <array of mandatory non-key property names (see CIM
1151         Requirements)>;
1152 }
1153 &smShowInstances ( "CIM_ProcessorCapabilities", "CIM_MemberOfCollection", NULL, NULL,
1154     $containerInstance.getObjectPath(), NULL, NULL, #propertylist[] );
1155 &smEnd;
```

### 1156 6.8.2.2 Show Command Form for a Single Instance Target

1157 This command form is used to show a single instance of CIM\_ProcessorCapabilities.

#### 1158 6.8.2.2.1 Command Form

```
1159 show <CIM_ProcessorCapabilities single instance>
```

#### 1160 6.8.2.2.2 CIM Requirements

1161 See CIM\_ProcessorCapabilities in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
1162 properties.

1163 **6.8.2.2.3 Behavior Requirements**1164 **6.8.2.2.3.1 Preconditions**

1165 In this section `$instance` represents the targeted instance of `CIM_ProcessorCapabilities`.

```
1166 $instance=<CIM_ProcessorCapabilities single instance>;
```

1167 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false

1168 **6.8.2.2.3.2 Pseudo Code**

```
1169 #propertylist[] = NULL;
1170 if ( false == #all)
1171 {
1172     #propertylist[] = <array of mandatory non-key property names (see CIM
1173     Requirements)>;
1174 }
1175 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1176 &smEnd;
```

1177 **6.9 CIM\_ProcessorCore**

1178 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1179 Table 9 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1180 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1181 target. Table 9 is for informational purposes only; in case of a conflict between Table 9 and requirements  
 1182 detailed in the following sections, the text detailed in the following sections supersedes the information in  
 1183 Table 9.

1184 **Table 9 – Command Verb Requirements for CIM\_ProcessorCore**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.9.2.
Set	May	See 6.9.3.
Show	Shall	See 6.9.4.
Start	May	See 6.9.5.
Stop	May	See 6.9.6.

1185 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, and `load`.

1186 **6.9.1 Ordering of Results**

1187 When results are returned for multiple instances of `CIM_ProcessorCore`, implementations shall utilize the  
 1188 following algorithm to produce the natural (that is, default) ordering:

- 1189
- Results for `CIM_ProcessorCore` are unordered; therefore, no algorithm is defined.

## 1190 6.9.2 Reset

1191 This section describes how to implement the `reset` verb when applied to an instance of  
1192 `CIM_ProcessorCore`. Implementations may support the use of the `reset` verb with `CIM_ProcessorCore`.

### 1193 6.9.2.1 Command Form

```
1194 reset <CIM_ProcessorCore single instance>
```

### 1195 6.9.2.2 CIM Requirements

```
1196 uint16 EnabledState;  
1197 uint16 RequestedState;  
1198 uint32 CIM_ProcessorCore.RequestStateChange (  
1199     [IN] uint16 RequestedState,  
1200     [OUT] REF CIM_ConcreteJob Job,  
1201     [IN] datetime TimeoutPeriod );
```

### 1202 6.9.2.3 Behavior Requirements

#### 1203 6.9.2.3.1 Preconditions

1204 In this section `$instance` represents the targeted instance of `CIM_ProcessorCore`.

```
1205 $instance=<CIM_ProcessorCore single instance>;
```

#### 1206 6.9.2.3.1.2 Pseudo Code

```
1207 &smResetRSC ( $instance.GetObjectPath() );  
1208 &smEnd;
```

## 1209 6.9.3 Set

1210 This section describes how to implement the `set` verb when it is applied to an instance of  
1211 `CIM_ProcessorCore`. Implementations may support the use of the `set` verb with `CIM_ProcessorCore`.

1212 The `set` verb is used to modify descriptive properties of the `CIM_ProcessorCore` instance.

### 1213 6.9.3.1 General Usage of Set for a Single Property

1214 This command form corresponds to the general usage of the `set` verb to modify a single property of a  
1215 target instance. This is the most common case.

1216 The requirement for supporting modification of a property using this command form shall be equivalent to  
1217 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
1218 in the [CPU Profile](#).

#### 1219 6.9.3.1.1 Command Form

```
1220 set <CIM_ProcessorCore single instance> <propertyname>=<propertyvalue>
```

#### 1221 6.9.3.1.2 CIM Requirements

1222 See `CIM_ProcessorCore` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
1223 properties.

1224 **6.9.3.1.3 Behavior Requirements**

```

1225 $instance=<CIM_ProcessorCore single instance>
1226 #propertyName[] = {<propertyname>};
1227 #propertyValues[] = {<propertyvalue>};
1228 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1229 &smEnd;

```

1230 **6.9.3.2 General Usage of Set for Multiple Properties**

1231 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a  
 1232 target instance where there is not an explicit relationship between the properties. This is the most  
 1233 common case.

1234 The requirement for supporting modification of a property using this command form shall be equivalent to  
 1235 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
 1236 in the [CPU Profile](#).

1237 **6.9.3.2.1 Command Form**

```

1238 set <CIM_ProcessorCore single instance> <propertyname1>=<propertyvalue1>
1239 <propertynamen>=<propertyvaluen>

```

1240 **6.9.3.2.2 CIM Requirements**

1241 See `CIM_ProcessorCore` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
 1242 properties.

1243 **6.9.3.2.3 Behavior Requirements**

```

1244 $instance=<CIM_ProcessorCore single instance>
1245 #propertyName[] = {<propertyname>};
1246 for #i < n
1247 {
1248     #propertyName[#i] = <propertyname#i>
1249     #propertyValues[#i] = <propertyvalue#i>
1250 }
1251 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1252 &smEnd;

```

1253 **6.9.4 Show**

1254 This section describes how to implement the `show` verb when applied to an instance of  
 1255 `CIM_ProcessorCore`. Implementations shall support the use of the `show` verb with `CIM_ProcessorCore`.

1256 **6.9.4.1 Show Command Form for Multiple Instances Target**

1257 This command form is used to show many instances of `CIM_ProcessorCore`.

1258 **6.9.4.1.1 Command Form**

```

1259 show <CIM_ProcessorCore multiple instances>

```

1260 **6.9.4.1.2 CIM Requirements**

1261 See `CIM_ProcessorCore` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
 1262 properties.



### 1263 6.9.4.1.3 Behavior Requirements

#### 1264 6.9.4.1.3.1 Preconditions

1265 In this section `$containerInstance` represents the instance of `CIM_Processor` which represents the  
1266 container system and is associated to the targeted instances of `CIM_ProcessorCore` through the  
1267 `CIM_ConcreteComponent` association.

1268 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

#### 1269 6.9.4.1.3.2 Pseudo Code

```
1270 #propertylist[] = NULL;
1271 if ( false == #all)
1272 {
1273     #propertylist[] = <array of mandatory non-key property names (see CIM
1274     Requirements)>;
1275 }
1276 &smShowInstances ( "CIM_ProcessorCore", "CIM_ConcreteComponent", NULL, NULL,
1277     $containerInstance.getObjectPath(), NULL, NULL, #propertylist[] );
1278 &smEnd;
```

### 1279 6.9.4.2 Show Command Form for a Single Instance Target

1280 This command form is used to show a single instance of `CIM_ProcessorCore`.

#### 1281 6.9.4.2.1 Command Form

```
1282 show <CIM_ProcessorCore single instance>
```

#### 1283 6.9.4.2.2 CIM Requirements

1284 See `CIM_ProcessorCore` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
1285 properties.

### 1286 6.9.4.2.3 Behavior Requirements

#### 1287 6.9.4.2.3.1 Preconditions

1288 In this section `$instance` represents the targeted instance of `CIM_ProcessorCore`.

```
1289 $instance=<CIM_ProcessorCore single instance>;
```

1290 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

#### 1291 6.9.4.2.3.2 Pseudo Code

```
1292 #propertylist[] = NULL;
1293 if ( false == #all)
1294 {
1295     #propertylist[] = <array of mandatory non-key property names (see CIM
1296     Requirements)>;
1297 }
1298 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1299 &smEnd;
```

## 1300 6.9.5 Start

1301 This section describes how to implement the `start` verb when applied to an instance of  
1302 `CIM_ProcessorCore`. Implementations may support the use of the `start` verb with `CIM_ProcessorCore`.

1303 **6.9.5.1 Command Form**1304 `start <CIM_ProcessorCore single instance>`1305 **6.9.5.2 CIM Requirements**

```
1306 uint16 EnabledState;
1307 uint16 RequestedState;
1308 uint32 CIM_ProcessorCore.RequestStateChange (
1309     [IN] uint16 RequestedState,
1310     [OUT] REF CIM_ConcreteJob Job,
1311     [IN] datetime TimeoutPeriod );
```

1312 **6.9.5.3 Behavior Requirements**1313 **6.9.5.3.1.1 Preconditions**1314 In this section `$instance` represents the targeted instance of `CIM_ProcessorCore`.1315 `$instance=<CIM_ProcessorCore single instance>;`1316 **6.9.5.3.1.2 Pseudo Code**

```
1317 &smStartRSC ( $instance.GetObjectPath() );
1318 &smEnd;
```

1319 **6.9.6 Stop**

1320 This section describes how to implement the `stop` verb when applied to an instance of  
 1321 `CIM_ProcessorCore`. Implementations may support the use of the `stop` verb with `CIM_ProcessorCore`.

1322 **6.9.6.1 Command Form**1323 `stop <CIM_ProcessorCore single instance>`1324 **6.9.6.1.1 CIM Requirements**

```
1325 uint16 EnabledState;
1326 uint16 RequestedState;
1327 uint32 CIM_ProcessorCore.RequestStateChange (
1328     [IN] uint16 RequestedState,
1329     [OUT] REF CIM_ConcreteJob Job,
1330     [IN] datetime TimeoutPeriod );
```

1331 **6.9.6.1.2 Behavior Requirements**1332 **6.9.6.1.2.1 Preconditions**1333 In this section `$instance` represents the targeted instance of `CIM_ProcessorCore`.1334 `$instance=<CIM_ProcessorCore single instance>;`1335 **6.9.6.1.2.2 Pseudo Code**

```
1336 &smStopRSC ( $instance.GetObjectPath() );
1337 &smEnd;
```

1338 **6.10 CIM\_SystemDevice**1339 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1340 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1341 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1342 target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and  
 1343 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1344 information in Table 10.

1345 **Table 10 – Command Verb Requirements for CIM\_SystemDevice**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.10.2.
Start	Not supported	
Stop	Not supported	

1346 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,  
 1347 `reset`, `set`, `start`, and `stop`.

### 1348 **6.10.1 Ordering of Results**

1349 When results are returned for multiple instances of `CIM_SystemDevice`, implementations shall utilize the  
 1350 following algorithm to produce the natural (that is, default) ordering:

- 1351 • Results for `CIM_SystemDevice` are unordered; therefore, no algorithm is defined.

### 1352 **6.10.2 Show**

1353 This section describes how to implement the `show` verb when applied to an instance of  
 1354 `CIM_SystemDevice`. Implementations shall support the use of the `show` verb with `CIM_SystemDevice`.

#### 1355 **6.10.2.1 Show Command Form for Multiple Instances Target – CIM\_ComputerSystem Reference**

1356 This command form is used to show many instances of `CIM_SystemDevice`. This command form  
 1357 corresponds to a `show` command issued against the instance of `CIM_SystemDevice` where only one  
 1358 reference is specified and the reference is to the scoping instance of `CIM_ComputerSystem`.

##### 1359 **6.10.2.1.1 Command Form**

1360 `show <CIM_SystemDevice multiple instances>`

##### 1361 **6.10.2.1.2 CIM Requirements**

1362 See `CIM_SystemDevice` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
 1363 properties.

### 1364 6.10.2.1.3 Behavior Requirements

#### 1365 6.10.2.1.3.1 Preconditions

1366 In this section `$instance` represents the instance of a `CIM_ComputerSystem`, which is referenced by  
1367 `CIM_SystemDevice`.

#### 1368 6.10.2.1.3.2 Pseudo Code

```
1369 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.GetObjectPath() );  
1370 &smEnd;
```

### 1371 6.10.2.2 Show Command Form for a Single Instance Target – CIM\_Processor Reference

1372 This command form is used to show a single instance of `CIM_SystemDevice`. This command form  
1373 corresponds to a `show` command issued against a single instance of `CIM_SystemDevice`, where only one  
1374 reference is specified and the reference is to the instance of `CIM_Processor`.

#### 1375 6.10.2.2.1 Command Form

```
1376 show <CIM_SystemDevice single instance>
```

#### 1377 6.10.2.2.2 CIM Requirements

1378 See `CIM_SystemDevice` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
1379 properties.

### 1380 6.10.2.2.3 Behavior Requirements

#### 1381 6.10.2.2.3.1 Preconditions

1382 In this section `$instance` represents the instance of `CIM_Processor` which is referenced by  
1383 `CIM_SystemDevice`.

#### 1384 6.10.2.2.3.2 Pseudo Code

```
1385 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.GetObjectPath() );  
1386 &smEnd;
```

### 1387 6.10.2.3 Show Command Form for a Single Instance Target – Both References

1388 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
1389 a `show` command issued against `CIM_SystemDevice` where both references are specified and therefore  
1390 the desired instance is unambiguously identified.

#### 1391 6.10.2.3.1 Command Form

```
1392 show <CIM_SystemDevice single instance>
```

#### 1393 6.10.2.3.2 CIM Requirements

1394 See `CIM_SystemDevice` in the “CIM Elements” section of the [CPU Profile](#) for the list of mandatory  
1395 properties.

**1396 6.10.2.3.3 Behavior Requirements****1397 6.10.2.3.3.1 Preconditions**

1398 In this section `$instanceA` represents the referenced instance of `CIM_Processor` through  
1399 `CIM_SystemDevice` association. `$instanceB` represents the instance of `CIM_ComputerSystem` which is  
1400 referenced by `CIM_SystemDevice`.

**1401 6.10.2.3.3.2 Pseudo Code**

```
1402 &smShowAssociationInstance ( "CIM_SystemDevice", $instanceA.getObjectPath(),  
1403     $instanceB.getObjectPath() );  
1404 &smEnd;
```

1405

1406  
1407  
1408  
1409  
1410

**ANNEX A**  
(informative)

**Change Log**

Version	Date	Author	Description
1.0.0	2009-06-04		DMTF Standard Release

1411