



1  
2  
3  
4

**Document Number: DSP0812**

**Date: 2009-07-14**

**Version: 1.0.0**

5 **Physical Asset Profile SM CLP Mapping**  
6 **Specification**

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

10

11 Copyright notice

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

13 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems  
14 management and interoperability. Members and non-members may reproduce DMTF specifications and  
15 documents, provided that correct attribution is given. As DMTF specifications may be revised from time to  
16 time, the particular version and release date should always be 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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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.

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

33

# CONTENTS

34	Foreword .....	5
35	Introduction .....	6
36	1 Scope .....	7
37	2 Normative References.....	7
38	2.1 Approved References .....	7
39	2.2 Other References.....	7
40	3 Terms and Definitions.....	7
41	4 Symbols and Abbreviated Terms.....	8
42	5 Recipes.....	9
43	6 Mappings.....	9
44	6.1 CIM_Card.....	9
45	6.2 CIM_Chassis.....	12
46	6.3 CIM_Chip .....	16
47	6.4 CIM_ComputerSystemPackage .....	18
48	6.5 CIM_ConfigurationCapacity .....	21
49	6.6 CIM_ConnectedTo.....	22
50	6.7 CIM_Container.....	24
51	6.8 CIM_ElementCapabilities .....	27
52	6.9 CIM_ElementCapacity .....	29
53	6.10 CIM_PackageInConnector.....	31
54	6.11 CIM_PhysicalAssetCapabilities .....	33
55	6.12 CIM_PhysicalComponent .....	35
56	6.13 CIM_PhysicalConnector .....	38
57	6.14 CIM_PhysicalFrame.....	41
58	6.15 CIM_PhysicalMemory .....	44
59	6.16 CIM_PhysicalPackage .....	46
60	6.17 CIM_Rack .....	50
61	6.18 CIM_Realizes.....	53
62	6.19 CIM_Slot .....	55
63	6.20 CIM_SystemPackaging.....	58
64	ANNEX A (informative) Change Log .....	61
65		

## 66 Tables

67	Table 1 – Command Verb Requirements for CIM_Card.....	9
68	Table 2 – Command Verb Requirements for CIM_Chassis.....	13
69	Table 3 – Command Verb Requirements for CIM_Chip .....	16
70	Table 4 – Command Verb Requirements for CIM_ComputerSystemPackage .....	18
71	Table 5 – Command Verb Requirements for CIM_ConfigurationCapacity .....	21
72	Table 6 – Command Verb Requirements for CIM_ConnectedTo.....	23
73	Table 7 – Command Verb Requirements for CIM_Container.....	25
74	Table 8 – Command Verb Requirements for CIM_ElementCapabilities .....	27
75	Table 9 – Command Verb Requirements for CIM_ElementCapacity .....	29
76	Table 10 – Command Verb Requirements for CIM_PackageInConnector.....	31
77	Table 11 – Command Verb Requirements for CIM_PhysicalAssetCapabilities .....	34
78	Table 12 – Command Verb Requirements for CIM_PhysicalComponent .....	36
79	Table 13 – Command Verb Requirements for CIM_PhysicalConnector .....	38

80	Table 14 – Command Verb Requirements for CIM_PhysicalFrame.....	41
81	Table 15 – Command Verb Requirements for CIM_PhysicalMemory .....	44
82	Table 16 – Command Verb Requirements for CIM_PhysicalPackage .....	47
83	Table 17 – Command Verb Requirements for CIM_Rack .....	50
84	Table 18 – Command Verb Requirements for CIM_Realizes.....	53
85	Table 19 – Command Verb Requirements for CIM_Slot .....	55
86	Table 20 – Command Verb Requirements for CIM_SystemPackaging .....	58
87		

88

## Foreword

89 The *Physical Asset Profile SM CLP Mapping Specification* (DSP0812) was prepared by the Server  
90 Management Working Group.

### 91 **Conventions**

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

### 94 **Acknowledgements**

95 The authors wish to acknowledge the following participants from the DMTF Server Management Working  
96 Group:

- 97 • Khachatur Papanyan – Dell Inc.
- 98 • Jon Hass – Dell Inc.
- 99 • Jianwen Yin – Dell Inc.
- 100 • Jeff Hilland – HP
- 101 • Christina Shaw – HP
- 102 • Aaron Merkin – IBM
- 103 • Perry Vincent – Intel
- 104 • John Leung – Intel

105

106

## Introduction

107 This document defines the SM CLP mapping for CIM elements described in the [Physical Asset Profile](#).  
108 The information in this specification, combined with the [SM CLP-to-CIM Common Mapping Specification](#)  
109 [1.0](#), is intended to be sufficient to implement SM CLP commands relevant to the classes, properties, and  
110 methods described in the [Physical Asset Profile](#) using CIM operations.

111 The target audience for this specification is implementers of the SM CLP support for the [Physical Asset](#)  
112 [Profile](#).

# 113 Physical Asset Profile SM CLP Mapping Specification

## 114 1 Scope

115 This specification contains the requirements for an implementation of the SM CLP to provide access to  
116 and implement the behaviors of the [Physical Asset Profile](#).

## 117 2 Normative References

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

### 121 2.1 Approved References

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

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

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

### 128 2.2 Other References

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

## 131 3 Terms and Definitions

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

### 133 3.1

#### 134 **can**

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

### 136 3.2

#### 137 **cannot**

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

### 139 3.3

#### 140 **conditional**

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

### 143 3.4

#### 144 **mandatory**

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

- 147 **3.5**  
148 **may**  
149 indicates a course of action permissible within the limits of the document
- 150 **3.6**  
151 **need not**  
152 indicates a course of action permissible within the limits of the document
- 153 **3.7**  
154 **optional**  
155 indicates a course of action permissible within the limits of the document
- 156 **3.8**  
157 **shall**  
158 indicates requirements to be followed strictly in order to conform to the document and from which no  
159 deviation is permitted
- 160 **3.9**  
161 **shall not**  
162 indicates requirements to be followed strictly in order to conform to the document and from which no  
163 deviation is permitted
- 164 **3.10**  
165 **should**  
166 indicates that among several possibilities, one is recommended as particularly suitable, without  
167 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 168 **3.11**  
169 **should not**  
170 indicates that a certain possibility or course of action is deprecated but not prohibited

## 171 **4 Symbols and Abbreviated Terms**

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

- 173 **4.1**  
174 **CIM**  
175 Common Information Model
- 176 **4.2**  
177 **CLP**  
178 Command Line Protocol
- 179 **4.3**  
180 **DMTF**  
181 Distributed Management Task Force
- 182 **4.4**  
183 **IETF**  
184 Internet Engineering Task Force



185 **4.5**  
 186 **SM**  
 187 Server Management

188 **4.6**  
 189 **SMI**  
 190 Storage Management Initiative

191 **4.7**  
 192 **SNIA**  
 193 Storage Networking Industry Association

194 **5 Recipes**

195 The following is a list of the common recipes used by the mappings in this specification. For a definition of  
 196 each recipe, see [DSP0216](#).

- 197 • smShowInstance
- 198 • smShowInstances
- 199 • smShowAssociationInstance
- 200 • smShowAssociationInstances

201 This mapping does not define any recipes for local reuse.

202 **6 Mappings**

203 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in  
 204 the [Physical Asset Profile](#).

205 **6.1 CIM\_Card**

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

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

212 **Table 1 – Command Verb Requirements for CIM\_Card**

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

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

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

### 215 6.1.1 Ordering of Results

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

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

### 219 6.1.2 Show

220 This section describes how to implement the `show` verb when applied to an instance of CIM\_Card.  
221 Implementations shall support the use of the `show` verb with CIM\_Card.

#### 222 6.1.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container 223 Instance

224 This command form is used to show many instances of CIM\_Card when CIM\_PhysicalPackage is the  
225 container instance.

#### 226 6.1.2.2 Command Form

```
227 show <CIM_Card multiple instances>
```

#### 228 6.1.2.3 CIM Requirements

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

#### 231 6.1.2.4 Behavior Requirements

##### 232 6.1.2.4.1 Preconditions

233 In this section `$containerInstance` represents the instance of CIM\_PhysicalPackage and is  
234 associated to the targeted instances of CIM\_Card through the CIM\_Container association.

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

##### 236 6.1.2.4.2 Pseudo Code

```
237 #propertylist[] = NULL;
238 if ( false == #all) {
239     #propertylist[] = <array of mandatory non-key property names (see CIM
240         Requirements)>;
241 }
242 &smShowInstances ( "CIM_Card", "CIM_Container", $containerInstance.getInstancePath(),
243     #propertylist[] );
244 &smEnd;
```

245 **6.1.2.5 Show Command Form for Multiple Instances Target – CIM\_PhysicalConnector**  
 246 **Container Instance**

247 This command form is used to show many instances of CIM\_Card when CIM\_PhysicalConnector is the  
 248 container instance.

249 **6.1.2.5.1 Command Form**

```
250 show <CIM_Card multiple instances>
```

251 **6.1.2.5.2 CIM Requirements**

252 See CIM\_Card in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 253 properties.

254 **6.1.2.5.3 Behavior Requirements**

255 **6.1.2.5.3.1 Preconditions**

256 In this section \$containerInstance represents the instance of CIM\_PhysicalConnector and is  
 257 associated to the targeted instances of CIM\_Card through the CIM\_PackageInConnector association.

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

259 **6.1.2.5.3.2 Pseudo Code**

```
260 #propertylist[] = NULL;
261 if ( false == #all) {
262     #propertylist[] = <array of mandatory non-key property names (see CIM
263     Requirements)>;
264 }
265 &smShowInstances ( "CIM_Card", "CIM_PackageInConnector",
266     $containerInstance.getInstancePath(), #propertylist[] );
267 &smEnd;
```

268 **6.1.2.6 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection**  
 269 **Container Instance**

270 This command form is used to show many instances of CIM\_Card when CIM\_ConcreteCollection is the  
 271 container instance.

272 **6.1.2.6.1 Command Form**

```
273 show <CIM_Card multiple instances>
```

274 **6.1.2.6.2 CIM Requirements**

275 See CIM\_Card in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 276 properties.

277 **6.1.2.6.3 Behavior Requirements**

278 **6.1.2.6.3.1 Preconditions**

279 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection and is  
 280 associated to the targeted instances of CIM\_Card through the CIM\_MemberOfCollection association.

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

282 **6.1.2.6.3.2 Pseudo Code**

```

283 #propertylist[] = NULL;
284 if ( false == #all) {
285     #propertylist[] = <array of mandatory non-key property names (see CIM
286         Requirements)>;
287 }
288 &smShowInstances ( "CIM_Card", "CIM_MemberOfCollection",
289     $containerInstance.getInstancePath(), #propertylist[] );
290 &smEnd;

```

291 **6.1.2.7 Show Command Form for a Single Instance Target**

292 This command form is used to show a single instance of CIM\_Card.

293 **6.1.2.7.1 Command Form**

```
294 show <CIM_Card single instance>
```

295 **6.1.2.7.2 CIM Requirements**

296 See CIM\_Card in the "CIM Elements" section of the [Physical Asset Profile](#) for the list of mandatory  
297 properties.

298 **6.1.2.7.3 Behavior Requirements**299 **6.1.2.7.3.1 Preconditions**

300 In this section \$instance represents the targeted instance of CIM\_Card.

```
301 $instance=<CIM_Card single instance>;
```

302 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

303 **6.1.2.7.3.2 Pseudo Code**

```

304 #propertylist[] = NULL;
305 if ( false == #all) {
306     #propertylist[] = <array of mandatory non-key property names (see CIM
307         Requirements)>;
308 }
309 &smShowInstance ( $instance, #propertylist[] );
310 &smEnd;

```

311 **6.2 CIM\_Chassis**

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

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

318

**Table 2 – Command Verb Requirements for CIM\_Chassis**

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	

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

321 **6.2.1 Ordering of Results**

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

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

325 **6.2.2 Show**

326 This section describes how to implement the show verb when applied to an instance of CIM\_Chassis.  
 327 Implementations shall support the use of the show verb with CIM\_Chassis.

328 **6.2.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container**  
 329 **Instance**

330 This command form is used to show many instances of CIM\_Chassis when CIM\_PhysicalPackage is the  
 331 container instance.

332 **6.2.2.1.1 Command Form**

333 `show <CIM_Chassis multiple instances>`

334 **6.2.2.1.2 CIM Requirements**

335 See CIM\_Chassis in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 336 properties.

337 **6.2.2.1.3 Behavior Requirements**

338 **6.2.2.1.3.1 Preconditions**

339 In this section \$containerInstance represents the instance of CIM\_PhysicalPackage and is  
 340 associated to the targeted instances of CIM\_Chassis through the CIM\_Container association.

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

342 **6.2.2.1.3.2 Pseudo Code**

```

343 #propertylist[] = NULL;
344 if ( false == #all) {
345     #propertylist[] = <array of mandatory non-key property names (see CIM
346         Requirements)>;
347 }
348 &smShowInstances ( "CIM_Chassis", "CIM_Container",
349     $containerInstance.getInstancePath(), #propertylist[] );
350 &smEnd;

```

351 **6.2.2.2 Show Command Form for Multiple Instances Target – CIM\_PhysicalConnector  
352 Container Instance**

353 This command form is used to show many instances of CIM\_Chassis when CIM\_PhysicalConnector is  
354 the container instance.

355 **6.2.2.2.1 Command Form**

```
356 show <CIM_Chassis multiple instances>
```

357 **6.2.2.2.2 CIM Requirements**

358 See CIM\_Chassis in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
359 properties.

360 **6.2.2.2.3 Behavior Requirements**361 **6.2.2.2.3.1 Preconditions**

362 In this section \$containerInstance represents the instance of CIM\_PhysicalConnector and is  
363 associated to the targeted instances of CIM\_Chassis through the CIM\_PackageInConnector association.

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

365 **6.2.2.2.3.2 Pseudo Code**

```

366 #propertylist[] = NULL;
367 if ( false == #all) {
368     #propertylist[] = <array of mandatory non-key property names (see CIM
369         Requirements)>;
370 }
371 &smShowInstances ( "CIM_Chassis", "CIM_PackageInConnector",
372     $containerInstance.getInstancePath(), #propertylist[] );
373 &smEnd;

```

374 **6.2.2.3 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection  
375 Container Instance**

376 This command form is used to show many instances of CIM\_Chassis when CIM\_ConcreteCollection is  
377 the container instance.

378 **6.2.2.3.1 Command Form**

```
379 show <CIM_Chassis multiple instances>
```

### 380 6.2.2.3.2 CIM Requirements

381 See CIM\_Chassis in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
382 properties.

### 383 6.2.2.3.3 Behavior Requirements

#### 384 6.2.2.3.3.1 Preconditions

385 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection and is  
386 associated to the targeted instances of CIM\_Chassis through the CIM\_MemberOfCollection association.

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

#### 388 6.2.2.3.3.2 Pseudo Code

```
389 #propertylist[] = NULL;
390 if ( false == #all) {
391     #propertylist[] = <array of mandatory non-key property names (see CIM
392         Requirements)>;
393 }
394 &smShowInstances ( "CIM_Chassis", "CIM_MemberOfCollection",
395     $containerInstance.getInstancePath(), #propertylist[] );
396 &smEnd;
```

### 397 6.2.2.4 Show Command Form for a Single Instance Target

398 This command form is used to show a single instance of CIM\_Chassis.

#### 399 6.2.2.4.1 Command Form

```
400 show <CIM_Chassis single instance>
```

#### 401 6.2.2.4.2 CIM Requirements

402 See CIM\_Chassis in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
403 properties.

#### 404 6.2.2.4.3 Behavior Requirements

##### 405 6.2.2.4.3.1 Preconditions

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

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

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

#### 409 6.2.2.4.3.2 Pseudo Code

```
410 #propertylist[] = NULL;
411 if ( false == #all) {
412     #propertylist[] = <array of mandatory non-key property names (see CIM
413         Requirements)>;
414 }
415 &smShowInstance ( $instance, #propertylist[] );
416 &smEnd;
```

417 **6.3 CIM\_Chip**418 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

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

424 **Table 3 – Command Verb Requirements for CIM\_Chip**

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 6.3.2.
Start	Not supported	
Stop	Not supported	

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

427 **6.3.1 Ordering of Results**

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

- 430 • Results for `CIM_Chip` are unordered; therefore, no algorithm is defined.

431 **6.3.2 Show**

432 This section describes how to implement the `show` verb when applied to an instance of `CIM_Chip`.  
 433 Implementations shall support the use of the `show` verb with `CIM_Chip`.

434 **6.3.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container Instance**

436 This command form is used to show many instances of `CIM_Chip` when `CIM_PhysicalPackage` is the  
 437 container instance.

438 **6.3.2.1.1 Command Form**439 `show <CIM_Chip multiple instances>`440 **6.3.2.1.2 CIM Requirements**

441 See `CIM_Chip` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 442 properties.



### 443 6.3.2.1.3 Behavior Requirements

#### 444 6.3.2.1.3.1 Preconditions

445 In this section `$containerInstance` represents the instance of `CIM_PhysicalPackage` and is  
446 associated to the targeted instances of `CIM_Chip` through the `CIM_Container` association.

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

#### 448 6.3.2.1.3.2 Pseudo Code

```
449 #propertylist[] = NULL;
450 if ( false == #all) {
451     #propertylist[] = <array of mandatory non-key property names (see CIM
452         Requirements)>;
453 }
454 &smShowInstances ( "CIM_Chip", "CIM_Container", $containerInstance.getInstancePath(),
455     #propertylist[] );
456 &smEnd;
```

### 457 6.3.2.2 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection 458 Container Instance

459 This command form is used to show many instances of `CIM_Chip` when `CIM_ConcreteCollection` is the  
460 container instance.

#### 461 6.3.2.2.1 Command Form

```
462 show <CIM_Chip multiple instances>
```

#### 463 6.3.2.2.2 CIM Requirements

464 See `CIM_Chip` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
465 properties.

### 466 6.3.2.2.3 Behavior Requirements

#### 467 6.3.2.2.3.1 Preconditions

468 In this section `$containerInstance` represents the instance of `CIM_ConcreteCollection` and is  
469 associated to the targeted instances of `CIM_Chip` through the `CIM_MemberOfCollection` association.

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

#### 471 6.3.2.2.3.2 Pseudo Code

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

### 480 6.3.2.3 Show Command Form for a Single Instance Target

481 This command form is used to show a single instance of `CIM_Chip`.

482 **6.3.2.3.1 Command Form**483 `show <CIM_Chip single instance>`484 **6.3.2.3.2 CIM Requirements**485 See CIM\_Chip in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
486 properties.487 **6.3.2.3.3 Behavior Requirements**488 **6.3.2.3.3.1 Preconditions**

489 In this section \$instance represents the targeted instance of CIM\_Chip.

490 `$instance=<CIM_Chip single instance>;`

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

492 **6.3.2.3.3.2 Pseudo Code**

```

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

```

500 **6.4 CIM\_ComputerSystemPackage**501 The cd, exit, help, and version verbs shall be supported as described in [DSP0216](#).

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

507 **Table 4 – Command Verb Requirements for CIM\_ComputerSystemPackage**

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 6.4.2.
Start	Not supported	
Stop	Not supported	

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

## 510 6.4.1 Ordering of Results

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

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

## 514 6.4.2 Show

515 This section describes how to implement the `show` verb when applied to an instance of  
516 CIM\_ComputerSystemPackage. Implementations shall support the use of the `show` verb with  
517 CIM\_ComputerSystemPackage.

### 518 6.4.2.1 Show Command Form for Multiple Instances Target – CIM\_ComputerSystem

519 This command form is used to show many instances of CIM\_ComputerSystemPackage. This command  
520 form corresponds to a `show` command issued against CIM\_ComputerSystemPackage where only one  
521 reference is specified and the reference is to an instance of CIM\_ComputerSystem.

#### 522 6.4.2.1.1 Command Form

```
523 show <CIM_ComputerSystemPackage multiple instances>
```

#### 524 6.4.2.1.2 CIM Requirements

525 See CIM\_ComputerSystemPackage in the “CIM Elements” section of the [Physical Asset Profile](#) for the list  
526 of mandatory properties and CIM classes that can be referenced.

#### 527 6.4.2.1.3 Behavior Requirements

##### 528 6.4.2.1.3.1 Preconditions

529 In this section `$instance` represents the instance of CIM\_ComputerSystem which is referenced by  
530 CIM\_ComputerSystemPackage.

531 Specifying the “-all” option does not change the output because the only property on the target instance is  
532 mandatory in “CIM Elements” section of the [Physical Asset Profile](#). Thus, no additional pseudo code is  
533 required to handle the option.

##### 534 6.4.2.1.3.2 Pseudo Code

```
535 &smShowAssociationInstances ( "CIM_ComputerSystemPackage",  
536     $instance.getInstancePath() );  
537 &smEnd;
```

### 538 6.4.2.2 Show Command Form for Multiple Instances – CIM\_PhysicalPackage Reference

539 This command form is used to show multiple instances of CIM\_ComputerSystemPackage. This command  
540 form corresponds to a `show` command issued against multiple instances of  
541 CIM\_ComputerSystemPackage where only one reference is specified and the reference is to an instance  
542 of CIM\_PhysicalPackage or a subclass of CIM\_PhysicalPackage.

#### 543 6.4.2.2.1 Command Form

```
544 show <CIM_ComputerSystemPackage multiple instances>
```

#### 545 6.4.2.2.2 CIM Requirements

546 See CIM\_ComputerSystemPackage in the “CIM Elements” section of the [Physical Asset Profile](#) for the list  
547 of mandatory properties and CIM classes that can be referenced.

#### 548 6.4.2.2.3 Behavior Requirements

##### 549 6.4.2.2.3.1 Preconditions

550 In this section `$instance` represents the instance of CIM\_PhysicalPackage or a subclass of  
551 CIM\_PhysicalPackage which is referenced by CIM\_ComputerSystemPackage.

552 Specifying the “-all” option does not change the output since the only property on the target instance is  
553 mandatory in the “CIM Elements” section of the [Physical Asset Profile](#). Thus, no additional pseudo code  
554 is required to handle the option.

##### 555 6.4.2.2.3.2 Pseudo Code

```
556 &smShowAssociationInstances ( "CIM_ComputerSystemPackage",  
557     $instance.getInstancePath() );  
558 &smEnd;
```

#### 559 6.4.2.3 Show Command Form for a Single Instance – Both References

560 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
561 a `show` command issued against CIM\_ComputerSystemPackage where both references are specified  
562 and therefore the desired instance is unambiguously identified.

##### 563 6.4.2.3.1 Command Form

```
564 show <CIM_ComputerSystemPackage single instance>
```

##### 565 6.4.2.3.2 CIM Requirements

566 See CIM\_ComputerSystemPackage in the “CIM Elements” section of the [Physical Asset Profile](#) for the list  
567 of mandatory properties and CIM classes that can be referenced.

##### 568 6.4.2.3.3 Behavior Requirements

##### 569 6.4.2.3.3.1 Preconditions

570 In this section `$instanceA` represents the referenced instance of CIM\_ComputerSystem through  
571 CIM\_ComputerSystemPackage association. `$instanceB` represents the instance of  
572 CIM\_PhysicalPackage or a subclass of CIM\_PhysicalPackage which is referenced by  
573 CIM\_ComputerSystemPackage.

574 Specifying the “-all” option does not change the output since the only property on the target instance is  
575 mandatory in the “CIM Elements” section of the [Physical Asset Profile](#). Thus, no additional pseudo code  
576 is required to handle the option.

##### 577 6.4.2.3.3.2 Pseudo Code

```
578 &smShowAssociationInstance ( "CIM_ComputerSystemPackage",  
579     $instanceA.getInstancePath(), $instanceB.getInstancePath() );  
580 &smEnd;
```

## 581 6.5 CIM\_ConfigurationCapacity

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

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

588 **Table 5 – Command Verb Requirements for CIM\_ConfigurationCapacity**

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.
Start	Not supported	
Stop	Not supported	

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

### 591 6.5.1 Ordering of Results

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

- 594 • Results for `CIM_ConfigurationCapacity` are unordered; therefore, no algorithm is defined.

### 595 6.5.2 Show

596 This section describes how to implement the `show` verb when applied to an instance of  
 597 `CIM_ConfigurationCapacity`. Implementations shall support the use of the `show` verb with  
 598 `CIM_ConfigurationCapacity`.

#### 599 6.5.2.1 Show Command Form for Multiple Instances Target

600 This command form is used to show many instances of `CIM_ConfigurationCapacity`.

##### 601 6.5.2.1.1 Command Form

602 `show <CIM_ConfigurationCapacity multiple instances>`

##### 603 6.5.2.1.2 CIM Requirements

604 See `CIM_ConfigurationCapacity` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
 605 mandatory properties.

### 606 6.5.2.1.3 Behavior Requirements

#### 607 6.5.2.1.3.1 Preconditions

608 In this section `$containerInstance` represents the instance of `CIM_ConcreteCollection`, and is  
609 associated to the targeted instances of `CIM_ConfigurationCapacity` through the `CIM_MemberOfCollection`  
610 association.

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

#### 612 6.5.2.1.3.2 Pseudo Code

```
613 #propertylist[] = NULL;
614 if ( false == #all) {
615     #propertylist[] = <array of mandatory non-key property names (see CIM
616         Requirements)>;
617 }
618 &smShowInstances ( "CIM_ConfigurationCapacity", "CIM_MemberOfCollection",
619     $containerInstance.getInstancePath(), #propertylist[] );
620 &smEnd;
```

### 621 6.5.2.2 Show Command Form for a Single Instance Target

622 This command form is used to show a single instance of `CIM_ConfigurationCapacity`.

#### 623 6.5.2.2.1 Command Form

```
624 show <CIM_ConfigurationCapacity single instance>
```

#### 625 6.5.2.2.2 CIM Requirements

626 See `CIM_ConfigurationCapacity` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
627 mandatory properties.

### 628 6.5.2.2.3 Behavior Requirements

#### 629 6.5.2.2.3.1 Preconditions

630 In this section `$instance` represents the targeted instance of `CIM_ConfigurationCapacity`.

```
631 $instance=<CIM_ConfigurationCapacity single instance>;
```

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

#### 633 6.5.2.2.3.2 Pseudo Code

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

## 641 6.6 CIM\_ConnectedTo

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

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

648 **Table 6 – Command Verb Requirements for CIM\_ConnectedTo**

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	

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

651 **6.6.1 Ordering of Results**

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

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

655 **6.6.2 Show**

656 This section describes how to implement the show verb when applied to an instance of  
 657 CIM\_ConnectedTo. Implementations shall support the use of the show verb with CIM\_ConnectedTo.

658 **6.6.2.1 Show Command Form for Multiple Instances Target**

659 This command form is used to show many instances of CIM\_ConnectedTo. This command form  
 660 corresponds to a show command issued against instances of CIM\_ConnectedTo where only one  
 661 reference is specified and the reference is to the instance of CIM\_PhysicalConnector or CIM\_Slot.

662 **6.6.2.1.1 Command Form**

663 `show <CIM_ConnectedTo multiple instances>`

664 **6.6.2.1.2 CIM Requirements**

665 See CIM\_ConnectedTo in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
 666 mandatory properties and CIM classes that can be referenced.

### 667 6.6.2.1.3 Behavior Requirements

#### 668 6.6.2.1.3.1 Preconditions

669 In this section `$instance` represents the instance of `CIM_PhysicalConnector` or `CIM_Slot` which is  
670 referenced by `CIM_ConnectedTo`.

#### 671 6.6.2.1.3.2 Pseudo Code

```
672 &smShowAssociationInstances ( "CIM_ConnectedTo", $instance.GetInstancePath() );  
673 &smEnd;
```

### 674 6.6.2.2 Show Command Form for a Single Instance – Both References

675 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
676 a `show` command issued against `CIM_ConnectedTo` where both references are specified and therefore  
677 the desired instance is unambiguously identified.

#### 678 6.6.2.2.1 Command Form

```
679 show <CIM_ConnectedTo single instance>
```

#### 680 6.6.2.2.2 CIM Requirements

681 See `CIM_ConnectedTo` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
682 mandatory properties and CIM classes that can be referenced.

#### 683 6.6.2.2.3 Behavior Requirements

##### 684 6.6.2.2.3.1 Preconditions

685 In this section `$instanceA` represents the referenced instance of `CIM_PhysicalConnector` or `CIM_Slot`  
686 through `CIM_ConnectedTo` association. `$instanceB` represents the other instance of  
687 `CIM_PhysicalConnector` or `CIM_Slot` which is referenced by `CIM_ConnectedTo`.

##### 688 6.6.2.2.3.2 Pseudo Code

```
689 &smShowAssociationInstance ( "CIM_ConnectedTo", $instanceA.GetInstancePath(),  
690     $instanceB.GetInstancePath() );  
691 &smEnd;
```

## 692 6.7 CIM\_Container

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

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



699

**Table 7 – Command Verb Requirements for CIM\_Container**

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 6.7.2.
Start	Not supported	
Stop	Not supported	

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

702 **6.7.1 Ordering of Results**

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

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

706 **6.7.2 Show**

707 This section describes how to implement the show verb when applied to an instance of CIM\_Container.  
 708 Implementations shall support the use of the show verb with CIM\_Container.

709 **6.7.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Reference**

710 This command form is used to show many instances of CIM\_Container. This command form corresponds  
 711 to a show command issued against instances of CIM\_Container where only one reference is specified  
 712 and the reference is to the instance of CIM\_PhysicalPackage or a subclass of CIM\_PhysicalPackage.

713 **6.7.2.1.1 Command Form**

714 `show <CIM_Container multiple instances>`

715 **6.7.2.1.2 CIM Requirements**

716 See CIM\_Container in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 717 properties and CIM classes that can be referenced.

718 **6.7.2.1.3 Behavior Requirements**

719 **6.7.2.1.3.1 Preconditions**

720 In this section \$instance represents the instance of CIM\_PhysicalPackage or a subclass of  
 721 CIM\_PhysicalPackage which is referenced by CIM\_Container.

### 722 6.7.2.1.3.2 Pseudo Code

```
723 &smShowAssociationInstances ( "CIM_Container", $instance.getInstancePath() );  
724 &smEnd;
```

## 725 6.7.2.2 Show Command Form for a Single Instance – CIM\_PhysicalElement Reference

726 This command form is used to show a single instance of CIM\_Container. This command form  
727 corresponds to a `show` command issued against a single instance of CIM\_Container where only one  
728 reference is specified and the reference is to the instance of a subclass of CIM\_PhysicalElement.

### 729 6.7.2.2.1 Command Form

```
730 show <CIM_Container single instance>
```

### 731 6.7.2.2.2 CIM Requirements

732 See CIM\_Container in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
733 properties and CIM classes that can be referenced.

### 734 6.7.2.2.3 Behavior Requirements

#### 735 6.7.2.2.3.1 Preconditions

736 In this section `$instance` represents the instance of a subclass of CIM\_PhysicalElement which is  
737 referenced by CIM\_Container.

#### 738 6.7.2.2.3.2 Pseudo Code

```
739 &smShowAssociationInstances ( "CIM_Container", $instance.getInstancePath() );  
740 &smEnd;
```

## 741 6.7.2.3 Show Command Form for a Single Instance – Both References

742 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
743 a `show` command issued against CIM\_Container where both references are specified and therefore the  
744 desired instance is unambiguously identified.

### 745 6.7.2.3.1 Command Form

```
746 show <CIM_Container single instance>
```

### 747 6.7.2.3.2 CIM Requirements

748 See CIM\_Container in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
749 properties and CIM classes that can be referenced.

### 750 6.7.2.3.3 Behavior Requirements

#### 751 6.7.2.3.3.1 Preconditions

752 In this section `$instanceA` represents the referenced instance of a subclass of CIM\_PhysicalElement  
753 through CIM\_Container association. `$instanceB` represents the instance of CIM\_PhysicalPackage or a  
754 subclass of CIM\_PhysicalPackage which is referenced by CIM\_Container.

#### 755 6.7.2.3.3.2 Pseudo Code

```
756 &smShowAssociationInstance ( "CIM_Container", $instanceA.getInstancePath(),  
757     $instanceB.getInstancePath() );  
758 &smEnd;
```

759 **6.8 CIM\_ElementCapabilities**

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

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

766 **Table 8 – 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 6.8.2.
Start	Not supported	
Stop	Not supported	

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

769 **6.8.1 Ordering of Results**

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

- 772 • Results for `CIM_ElementCapabilities` are unordered; therefore, no algorithm is defined.

773 **6.8.2 Show**

774 This section describes how to implement the `show` verb when applied to an instance of  
 775 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with  
 776 `CIM_ElementCapabilities`.

777 **6.8.2.1 Show Command Form for a Single Instance Target – CIM\_PhysicalElement Reference**

778 This command form is used to show a single instance of `CIM_ElementCapabilities`. This command form  
 779 corresponds to a `show` command issued against a single instance of `CIM_ElementCapabilities` where  
 780 only one reference is specified and the reference is to the instance of a subclass of  
 781 `CIM_PhysicalElement`.

782 **6.8.2.1.1 Command Form**

783 `show <CIM_ElementCapabilities single instance>`

784 **6.8.2.1.2 CIM Requirements**

785 See CIM\_ElementCapabilities in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
786 mandatory properties.

787 **6.8.2.1.3 Behavior Requirements**788 **6.8.2.1.3.1 Preconditions**

789 In this section `$instance` represents the instance of a subclass of CIM\_PhysicalElement which is  
790 referenced by CIM\_ElementCapabilities.

791 **6.8.2.1.3.2 Pseudo Code**

```
792 &smShowAssociationInstances ( "CIM_ElementCapabilities",
793     $instance.getInstancePath() );
794 &smEnd;
```

795 **6.8.2.2 Show Command Form for Multiple Instances –CIM\_PhysicalAssetCapabilities Reference**

797 This command form is used to show multiple instances of CIM\_ElementCapabilities. This command form  
798 corresponds to a `show` command issued against multiple instances of CIM\_ElementCapabilities where  
799 only one reference is specified and the reference is to the instance of CIM\_PhysicalAssetCapabilities.

800 **6.8.2.2.1 Command Form**

```
801 show <CIM_ElementCapabilities multiple instances>
```

802 **6.8.2.2.2 CIM Requirements**

803 See CIM\_ElementCapabilities in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
804 mandatory properties and CIM classes that can be referenced.

805 **6.8.2.2.3 Behavior Requirements**806 **6.8.2.2.3.1 Preconditions**

807 In this section `$instance` represents the instance of CIM\_PhysicalAssetCapabilities which is referenced  
808 by CIM\_ElementCapabilities.

809 **6.8.2.2.3.2 Pseudo Code**

```
810 &smShowAssociationInstances ( "CIM_ElementCapabilities",
811     $instance.getInstancePath() );
812 &smEnd;
```

813 **6.8.2.3 Show Command Form for a Single Instance – Both References**

814 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
815 a `show` command issued against CIM\_ElementCapabilities where both references are specified and  
816 therefore the desired instance is unambiguously identified.

817 **6.8.2.3.1 Command Form**

```
818 show <CIM_ElementCapabilities single instance>
```

819 **6.8.2.3.2 CIM Requirements**

820 See CIM\_ElementCapabilities in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
821 mandatory properties and CIM classes that can be referenced.

822 **6.8.2.3.3 Behavior Requirements**

823 **6.8.2.3.3.1 Preconditions**

824 In this section \$instanceA represents the referenced instance of a subclass of CIM\_PhysicalElement  
825 through CIM\_ElementCapabilities association. \$instanceB represents the instance of  
826 CIM\_PhysicalAssetCapabilities which is referenced by CIM\_ElementCapabilities.

827 **6.8.2.3.3.2 Pseudo Code**

```
828 &smShowAssociationInstance ( "CIM_ElementCapabilities",
829     $instanceA.getInstancePath(), $instanceB.getInstancePath() );
830 &smEnd;
```

831 **6.9 CIM\_ElementCapacity**

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

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

838 **Table 9 – Command Verb Requirements for CIM\_ElementCapacity**

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 6.9.2.
Start	Not supported	
Stop	Not supported	

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

841 **6.9.1 Ordering of Results**

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

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

## 845 6.9.2 Show

846 This section describes how to implement the `show` verb when applied to an instance of  
847 `CIM_ElementCapacity`. Implementations shall support the use of the `show` verb with  
848 `CIM_ElementCapacity`.

### 849 6.9.2.1 Show Command Form for Multiple Instances Target

850 This command form is used to show many instances of `CIM_ElementCapacity`. This command form  
851 corresponds to a `show` command issued against instances of `CIM_ElementCapacity` where only one  
852 reference is specified and the reference is to the instance of a subclass of `CIM_PhysicalElement`.

#### 853 6.9.2.1.1 Command Form

```
854 show <CIM_ElementCapacity multiple instances>
```

#### 855 6.9.2.1.2 CIM Requirements

856 See `CIM_ElementCapacity` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
857 mandatory properties and CIM classes that can be referenced.

#### 858 6.9.2.1.3 Behavior Requirements

##### 859 6.9.2.1.3.1 Preconditions

860 In this section `$instance` represents the instance of a subclass of `CIM_PhysicalElement` which is  
861 referenced by `CIM_ElementCapacity`.

##### 862 6.9.2.1.3.2 Pseudo Code

```
863 &smShowAssociationInstances ( "CIM_ElementCapacity", $instance.getInstancePath() );  
864 &smEnd;
```

### 865 6.9.2.2 Show Command Form for Multiple Instances – CIM\_ConfigurationCapacity Reference

866 This command form is used to show multiple instances of `CIM_ElementCapacity`. This command form  
867 corresponds to a `show` command issued against multiple instances of `CIM_ElementCapacity` where only  
868 one reference is specified and the reference is to the instance of `CIM_PhysicalAssetCapabilities`.

#### 869 6.9.2.2.1 Command Form

```
870 show <CIM_ElementCapacity multiple instances>
```

#### 871 6.9.2.2.2 CIM Requirements

872 See `CIM_ElementCapacity` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
873 mandatory properties and CIM classes that can be referenced.

#### 874 6.9.2.2.3 Behavior Requirements

##### 875 6.9.2.2.3.1 Preconditions

876 In this section `$instance` represents the instance of `CIM_ConfigurationCapacity` which is referenced by  
877 `CIM_ElementCapacity`.

##### 878 6.9.2.2.3.2 Pseudo Code

```
879 &smShowAssociationInstances ( "CIM_ElementCapacity", $instance.getInstancePath() );  
880 &smEnd;
```

881 **6.9.2.3 Show Command Form for a Single Instance – Both References**

882 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 883 a `show` command issued against `CIM_ElementCapacity` where both references are specified and  
 884 therefore the desired instance is unambiguously identified.

885 **6.9.2.3.1 Command Form**

886 `show <CIM_ElementCapacity single instance>`

887 **6.9.2.3.2 CIM Requirements**

888 See `CIM_ElementCapacity` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
 889 mandatory properties and CIM classes that can be referenced.

890 **6.9.2.3.3 Behavior Requirements**

891 **6.9.2.3.3.1 Preconditions**

892 In this section `$instanceA` represents the referenced instance of a subclass of `CIM_PhysicalElement`  
 893 through `CIM_ElementCapacity` association. `$instanceB` represents the instance of  
 894 `CIM_ConfigurationCapacity` which is referenced by `CIM_ElementCapacity`.

895 **6.9.2.3.3.2 Pseudo Code**

896 `&smShowAssociationInstance ( "CIM_ElementCapacity", $instanceA.getInstancePath(),`  
 897 `$instanceB.getInstancePath() );`  
 898 `&smEnd;`

899 **6.10 CIM\_PackageInConnector**

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

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

906 **Table 10 – Command Verb Requirements for CIM\_PackageInConnector**

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	

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

## 909 6.10.1 Ordering of Results

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

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

## 913 6.10.2 Show

914 This section describes how to implement the `show` verb when applied to an instance of  
915 CIM\_PackageInConnector. Implementations shall support the use of the `show` verb with  
916 CIM\_PackageInConnector.

### 917 6.10.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Reference

918 This command form is used to show many instances of CIM\_PackageInConnector. This command form  
919 corresponds to a `show` command issued against instances of CIM\_PackageInConnector where only one  
920 reference is specified and the reference is to the instance of CIM\_PhysicalPackage or a subclass of  
921 CIM\_PhysicalPackage.

#### 922 6.10.2.1.1 Command Form

```
923 show <CIM_PackageInConnector multiple instances>
```

#### 924 6.10.2.1.2 CIM Requirements

925 See CIM\_PackageInConnector in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
926 mandatory properties and CIM classes that can be referenced.

#### 927 6.10.2.1.3 Behavior Requirements

##### 928 6.10.2.1.3.1 Preconditions

929 In this section `$instance` represents the instance of CIM\_PhysicalPackage or a subclass of  
930 CIM\_PhysicalPackage which is referenced by CIM\_PackageInConnector.

##### 931 6.10.2.1.3.2 Pseudo Code

```
932 &smShowAssociationInstances ( "CIM_PackageInConnector", $instance.getInstancePath() );  
933 &smEnd;
```

### 934 6.10.2.2 Show Command Form for Multiple Instances – CIM\_PhysicalConnector Reference

935 This command form is used to show multiple instances of CIM\_PackageInConnector. This command form  
936 corresponds to a `show` command issued against multiple instances of CIM\_PackageInConnector where  
937 only one reference is specified and the reference is to the instance of CIM\_PhysicalConnector.

#### 938 6.10.2.2.1 Command Form

```
939 show <CIM_PackageInConnector multiple instances>
```

#### 940 6.10.2.2.2 CIM Requirements

941 See CIM\_PackageInConnector in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
942 mandatory properties and CIM classes that can be referenced.



### 943 6.10.2.2.3 Behavior Requirements

#### 944 6.10.2.2.3.1 Preconditions

945 In this section `$instance` represents the instance of `CIM_PhysicalConnector` which is referenced by  
946 `CIM_PackageInConnector`.

#### 947 6.10.2.2.3.2 Pseudo Code

```
948 &smShowAssociationInstances ( "CIM_PackageInConnector", $instance.GetInstancePath() );
949 &smEnd;
```

### 950 6.10.2.3 Show Command Form for a Single Instance – Both References

951 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
952 the `show` command issued against `CIM_PackageInConnector` where both references are specified and  
953 therefore the desired instance is unambiguously identified.

#### 954 6.10.2.3.1 Command Form

```
955 show <CIM_PackageInConnector single instance>
```

#### 956 6.10.2.3.2 CIM Requirements

957 See `CIM_PackageInConnector` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
958 mandatory properties and CIM classes that can be referenced.

### 959 6.10.2.3.3 Behavior Requirements

#### 960 6.10.2.3.3.1 Preconditions

961 In this section `$instanceA` represents the referenced instance of `CIM_PhysicalPackage` or a subclass of  
962 `CIM_PhysicalPackage` through `CIM_PackageInConnector` association. `$instanceB` represents the  
963 instance of `CIM_PhysicalConnector` which is referenced by `CIM_PackageInConnector`.

#### 964 6.10.2.3.3.2 Pseudo Code

```
965 &smShowAssociationInstance ( "CIM_PackageInConnector", $instanceA.GetInstancePath(),
966     $instanceB.GetInstancePath() );
967 &smEnd;
```

## 968 6.11 CIM\_PhysicalAssetCapabilities

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

970 Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
971 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
972 target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and  
973 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
974 information in Table 11.

975

Table 11 – Command Verb Requirements for CIM\_PhysicalAssetCapabilities

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 6.11.2.
Start	Not supported	
Stop	Not supported	

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

### 978 6.11.1 Ordering of Results

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

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

### 982 6.11.2 Show

983 This section describes how to implement the `show` verb when applied to an instance of  
 984 CIM\_PhysicalAssetCapabilities. Implementations shall support the use of the `show` verb with  
 985 CIM\_PhysicalAssetCapabilities.

#### 986 6.11.2.1 Show Command Form for Multiple Instances Target

987 This command form is used to show many instances of CIM\_PhysicalAssetCapabilities.

##### 988 6.11.2.1.1 Command Form

```
989 show <CIM_PhysicalAssetCapabilities multiple instances>
```

##### 990 6.11.2.1.2 CIM Requirements

991 See CIM\_PhysicalAssetCapabilities in the “CIM Elements” section of the [Physical Asset Profile](#) for the list  
 992 of mandatory properties.

##### 993 6.11.2.1.3 Behavior Requirements

###### 994 6.11.2.1.3.1 Preconditions

995 In this section `$containerInstance` represents the instance of CIM\_ConcreteCollection, and is  
 996 associated to the targeted instances of CIM\_PhysicalAssetCapabilities through the  
 997 CIM\_MemberOfCollection association.

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

### 999 6.11.2.1.3.2 Pseudo Code

```

1000 #propertylist[] = NULL;
1001 if ( false == #all) {
1002     #propertylist[] = <array of mandatory non-key property names (see CIM
1003         Requirements)>;
1004 }
1005 &smShowInstances ( "CIM_PhysicalAssetCapabilities", "CIM_MemberOfCollection",
1006     $containerInstance.getInstancePath(), #propertylist[] );
1007 &smEnd;

```

### 1008 6.11.2.2 Show Command Form for a Single Instance Target

1009 This command form is used to show a single instance of CIM\_PhysicalAssetCapabilities.

#### 1010 6.11.2.2.1 Command Form

```

1011 show <CIM_PhysicalAssetCapabilities single instance>

```

#### 1012 6.11.2.2.2 CIM Requirements

1013 See CIM\_PhysicalAssetCapabilities in the “CIM Elements” section of the [Physical Asset Profile](#) for the list  
 1014 of mandatory properties.

#### 1015 6.11.2.2.3 Behavior Requirements

##### 1016 6.11.2.2.3.1 Preconditions

1017 In this section \$instance represents the targeted instance of CIM\_PhysicalAssetCapabilities.

```

1018 $instance=<CIM_PhysicalAssetCapabilities single instance>;

```

1019 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

##### 1020 6.11.2.2.3.2 Pseudo Code

```

1021 #propertylist[] = NULL;
1022 if ( false == #all) {
1023     #propertylist[] = <array of mandatory non-key property names (see CIM
1024         Requirements)>;
1025 }
1026 &smShowInstance ( $instance, #propertylist[] );
1027 &smEnd;

```

## 1028 6.12 CIM\_PhysicalComponent

1029 The cd, exit, help, and version verbs shall be supported as described in [DSP0216](#).

1030 Table 12 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1031 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1032 target. Table 12 is for informational purposes only; in case of a conflict between Table 12 and  
 1033 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1034 information in Table 12.

1035

Table 12 – Command Verb Requirements for CIM\_PhysicalComponent

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 6.12.2.
Start	Not supported	
Stop	Not supported	

1036 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,  
 1037 `load`, `reset`, `set`, `start`, and `stop`.

### 1038 6.12.1 Ordering of Results

1039 When results are returned for multiple instances of `CIM_ElementCapabilities`, implementations shall  
 1040 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1041 • Results for `CIM_PhysicalComponent` are unordered; therefore, no algorithm is defined.

### 1042 6.12.2 Show

1043 This section describes how to implement the `show` verb when applied to an instance of  
 1044 `CIM_PhysicalComponent`. Implementations shall support the use of the `show` verb with  
 1045 `CIM_PhysicalComponent`.

#### 1046 6.12.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container 1047 Instance

1048 This command form is used to show many instances of `CIM_PhysicalComponent` when  
 1049 `CIM_PhysicalPackage` is the container instance.

##### 1050 6.12.2.1.1 Command Form

```
1051 show <CIM_PhysicalComponent multiple instances>
```

##### 1052 6.12.2.1.2 CIM Requirements

1053 See the “CIM Elements” section in the [Physical Asset Profile](#).

##### 1054 6.12.2.1.3 Behavior Requirements

###### 1055 6.12.2.1.3.1 Preconditions

1056 In this section `$containerInstance` represents the instance of `CIM_PhysicalPacakge` and is  
 1057 associated to the targeted instances of `CIM_PhysicalComponent` through the `CIM_Container` association.

1058 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

### 1059 6.12.2.1.3.2 Pseudo Code

```

1060 #propertylist[] = NULL;
1061 if ( false == #all) {
1062     #propertylist[] = <array of mandatory non-key property names (see CIM
1063         Requirements)>;
1064 }
1065 &smShowInstances ( "CIM_PhysicalComponent", "CIM_Container",
1066     $containerInstance.getInstancePath(), #propertylist[] );
1067 &smEnd;

```

### 1068 6.12.2.2 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection 1069 Container Instance

1070 This command form is used to show many instances of CIM\_PhysicalComponent when  
1071 CIM\_ConcreteCollection is the container instance.

#### 1072 6.12.2.2.1 Command Form

```
1073 show <CIM_PhysicalComponent multiple instances>
```

#### 1074 6.12.2.2.2 CIM Requirements

1075 See the “CIM Elements” section in the [Physical Asset Profile](#).

#### 1076 6.12.2.2.3 Behavior Requirements

##### 1077 6.12.2.2.3.1 Preconditions

1078 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection and is  
1079 associated to the targeted instances of CIM\_PhysicalComponent through the CIM\_MemberOfCollection  
1080 association.

1081 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

##### 1082 6.12.2.2.3.2 Pseudo Code

```

1083 #propertylist[] = NULL;
1084 if ( false == #all) {
1085     #propertylist[] = <array of mandatory non-key property names (see CIM
1086         Requirements)>;
1087 }
1088 &smShowInstances ( "CIM_PhysicalComponent", "CIM_MemberOfCollection",
1089     $containerInstance.getInstancePath(), #propertylist[] );
1090 &smEnd;

```

### 1091 6.12.2.3 Show Command Form for a Single Instance Target

1092 This command form is used to show a single instance of CIM\_PhysicalComponent.

#### 1093 6.12.2.3.1 Command Form

```
1094 show <CIM_PhysicalComponent single instance>
```

#### 1095 6.12.2.3.2 CIM Requirements

1096 See the “CIM Elements” section in the [Physical Asset Profile](#).

1097 **6.12.2.3.3 Behavior Requirements**1098 **6.12.2.3.3.1 Preconditions**

1099 In this section `$instance` represents the targeted instance of `CIM_PhysicalComponent`.

```
1100 $instance=<CIM_PhysicalComponent single instance>;
```

1101 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1102 **6.12.2.3.3.2 Pseudo Code**

```
1103 #propertylist[] = NULL;
1104 if ( false == #all) {
1105     #propertylist[] = <array of mandatory non-key property names (see CIM
1106         Requirements)>;
1107 }
1108 &smShowInstance ( $instance, #propertylist[] );
1109 &smEnd;
```

1110 **6.13 CIM\_PhysicalConnector**

1111 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1112 Table 13 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1113 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1114 target. Table 13 is for informational purposes only; in case of a conflict between Table 13 and  
 1115 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1116 information in Table 13.

1117 **Table 13 – Command Verb Requirements for CIM\_PhysicalConnector**

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 6.13.2.
Start	Not supported	
Stop	Not supported	

1118 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,  
 1119 `load`, `reset`, `set`, `start`, and `stop`.

1120 **6.13.1 Ordering of Results**

1121 When results are returned for multiple instances of `CIM_ElementCapabilities`, implementations shall  
 1122 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1123 • Results for `CIM_PhysicalConnector` are unordered; therefore, no algorithm is defined.

## 1124 **6.13.2 Show**

1125 This section describes how to implement the `show` verb when applied to an instance of  
1126 `CIM_PhysicalConnector`. Implementations shall support the use of the `show` verb with  
1127 `CIM_PhysicalConnector`.

### 1128 **6.13.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container Instance**

1130 This command form is used to show many instances of `CIM_PhysicalConnector` when  
1131 `CIM_PhysicalPackage` is the container instance.

#### 1132 **6.13.2.1.1 Command Form**

```
1133 show <CIM_PhysicalConnector multiple instances>
```

#### 1134 **6.13.2.1.2 CIM Requirements**

1135 See `CIM_PhysicalConnector` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1136 mandatory properties.

#### 1137 **6.13.2.1.3 Behavior Requirements**

##### 1138 **6.13.2.1.3.1 Preconditions**

1139 In this section `$containerInstance` represents the instance of `CIM_PhysicalPackage` and is  
1140 associated to the targeted instances of `CIM_PhysicalConnector` through the `CIM_Container` association.

1141 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

##### 1142 **6.13.2.1.3.2 Pseudo Code**

```
1143 #propertylist[] = NULL;
1144 if ( false == #all) {
1145     #propertylist[] = <array of mandatory non-key property names (see CIM
1146         Requirements)>;
1147 }
1148 &smShowInstances ( "CIM_PhysicalConnector", "CIM_Container",
1149     $containerInstance.getInstancePath(), #propertylist[] );
1150 &smEnd;
```

### 1151 **6.13.2.2 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection Container Instance**

1153 This command form is used to show many instances of `CIM_PhysicalConnector` when  
1154 `CIM_ConcreteCollection` is the container instance.

#### 1155 **6.13.2.2.1 Command Form**

```
1156 show <CIM_PhysicalConnector multiple instances>
```

#### 1157 **6.13.2.2.2 CIM Requirements**

1158 See `CIM_PhysicalConnector` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1159 mandatory properties.

1160 **6.13.2.2.3 Behavior Requirements**1161 **6.13.2.2.3.1 Preconditions**

1162 In this section `$containerInstance` represents the instance of `CIM_ConcreteCollection` and is  
 1163 associated to the targeted instances of `CIM_PhysicalConnector` through the `CIM_MemberOfCollection`  
 1164 association.

1165 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1166 **6.13.2.2.3.2 Pseudo Code**

```
1167 #propertylist[] = NULL;
1168 if ( false == #all) {
1169     #propertylist[] = <array of mandatory non-key property names (see CIM
1170         Requirements)>;
1171 }
1172 &smShowInstances ( "CIM_PhysicalConnector", "CIM_MemberOfCollection",
1173     $containerInstance.getInstancePath(), #propertylist[] );
1174 &smEnd;
```

1175 **6.13.2.3 Show Command Form for a Single Instance Target**

1176 This command form is used to show a single instance of `CIM_PhysicalConnector`.

1177 **6.13.2.3.1 Command Form**

```
1178 show <CIM_PhysicalConnector single instance>
```

1179 **6.13.2.3.2 CIM Requirements**

1180 See `CIM_PhysicalConnector` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
 1181 mandatory properties.

1182 **6.13.2.3.3 Behavior Requirements**1183 **6.13.2.3.3.1 Preconditions**

1184 In this section `$instance` represents the targeted instance of `CIM_PhysicalConnector`.

```
1185 $instance=<CIM_PhysicalConnector single instance>;
```

1186 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1187 **6.13.2.3.3.2 Pseudo Code**

```
1188 #propertylist[] = NULL;
1189 if ( false == #all) {
1190     #propertylist[] = <array of mandatory non-key property names (see CIM
1191         Requirements)>;
1192 }
1193 &smShowInstance ( $instance, #propertylist[] );
1194 &smEnd;
```



1195 **6.14 CIM\_PhysicalFrame**

1196 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1197 Table 14 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1198 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1199 target. Table 14 is for informational purposes only; in case of a conflict between Table 14 and  
 1200 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1201 information in Table 14.

1202 **Table 14 – Command Verb Requirements for CIM\_PhysicalFrame**

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 6.14.2.
Start	Not supported	
Stop	Not supported	

1203 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,  
 1204 `load`, `reset`, `set`, `start`, and `stop`.

1205 **6.14.1 Ordering of Results**

1206 When results are returned for multiple instances of `CIM_ElementCapabilities`, implementations shall  
 1207 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1208 • Results for `CIM_PhysicalFrame` are unordered; therefore, no algorithm is defined.

1209 **6.14.2 Show**

1210 This section describes how to implement the `show` verb when applied to an instance of  
 1211 `CIM_PhysicalFrame`. Implementations shall support the use of the `show` verb with `CIM_PhysicalFrame`.

1212 **6.14.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container**  
 1213 **Instance**

1214 This command form is used to show many instances of `CIM_PhysicalFrame` when `CIM_PhysicalPackage`  
 1215 is the container instance.

1216 **6.14.2.1.1 Command Form**

1217 `show <CIM_PhysicalFrame multiple instances>`

1218 **6.14.2.1.2 CIM Requirements**

1219 See `CIM_PhysicalFrame` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
 1220 mandatory properties.

1221 **6.14.2.1.3 Behavior Requirements**1222 **6.14.2.1.3.1 Preconditions**

1223 In this section `$containerInstance` represents the instance of `CIM_PhysicalPackage` and is  
 1224 associated to the targeted instances of `CIM_PhysicalFrame` through the `CIM_Container` association.

1225 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1226 **6.14.2.1.3.2 Pseudo Code**

```
1227 #propertylist[] = NULL;
1228 if ( false == #all) {
1229     #propertylist[] = <array of mandatory non-key property names (see CIM
1230         Requirements)>;
1231 }
1232 &smShowInstances ( "CIM_PhysicalFrame", "CIM_Container",
1233     $containerInstance.getInstancePath(), #propertylist[] );
1234 &smEnd;
```

1235 **6.14.2.2 Show Command Form for Multiple Instances Target – CIM\_PhysicalConnector  
1236 Container Instance**

1237 This command form is used to show many instances of `CIM_PhysicalFrame` when  
 1238 `CIM_PhysicalConnector` is the container instance.

1239 **6.14.2.2.1 Command Form**

```
1240 show <CIM_PhysicalFrame multiple instances>
```

1241 **6.14.2.2.2 CIM Requirements**

1242 See `CIM_PhysicalFrame` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
 1243 mandatory properties.

1244 **6.14.2.2.3 Behavior Requirements**1245 **6.14.2.2.3.1 Preconditions**

1246 In this section `$containerInstance` represents the instance of `CIM_PhysicalConnector` and is  
 1247 associated to the targeted instances of `CIM_PhysicalFrame` through the `CIM_PackageInConnector`  
 1248 association.

1249 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1250 **6.14.2.2.3.2 Pseudo Code**

```
1251 #propertylist[] = NULL;
1252 if ( false == #all) {
1253     #propertylist[] = <array of mandatory non-key property names (see CIM
1254         Requirements)>;
1255 }
1256 &smShowInstances ( "CIM_PhysicalFrame", "CIM_PackageInConnector",
1257     $containerInstance.getInstancePath(), #propertylist[] );
1258 &smEnd;
```

### 1259 **6.14.2.3 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection** 1260 **Container Instance**

1261 This command form is used to show many instances of CIM\_PhysicalFrame when  
1262 CIM\_ConcreteCollection is the container instance.

#### 1263 **6.14.2.3.1 Command Form**

```
1264 show <CIM_PhysicalFrame multiple instances>
```

#### 1265 **6.14.2.3.2 CIM Requirements**

1266 See CIM\_PhysicalFrame in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1267 mandatory properties.

#### 1268 **6.14.2.3.3 Behavior Requirements**

##### 1269 **6.14.2.3.3.1 Preconditions**

1270 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection and is  
1271 associated to the targeted instances of CIM\_PhysicalFrame through the CIM\_MemberOfCollection  
1272 association.

1273 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

##### 1274 **6.14.2.3.3.2 Pseudo Code**

```
1275 #propertylist[] = NULL;
1276 if ( false == #all) {
1277     #propertylist[] = <array of mandatory non-key property names (see CIM
1278         Requirements)>;
1279 }
1280 &smShowInstances ( "CIM_PhysicalFrame", "CIM_MemberOfCollection",
1281     $containerInstance.getInstancePath(), #propertylist[] );
1282 &smEnd;
```

### 1283 **6.14.2.4 Show Command Form for a Single Instance Target**

1284 This command form is used to show a single instance of CIM\_PhysicalFrame.

#### 1285 **6.14.2.4.1 Command Form**

```
1286 show <CIM_PhysicalFrame single instance>
```

#### 1287 **6.14.2.4.2 CIM Requirements**

1288 See CIM\_PhysicalFrame in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1289 mandatory properties.

#### 1290 **6.14.2.4.3 Behavior Requirements**

##### 1291 **6.14.2.4.3.1 Preconditions**

1292 In this section \$instance represents the targeted instance of CIM\_PhysicalFrame.

```
1293 $instance=<CIM_PhysicalFrame single instance>;
```

1294 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1295 **6.14.2.4.3.2 Pseudo Code**

```

1296 #propertylist[] = NULL;
1297 if ( false == #all) {
1298     #propertylist[] = <array of mandatory non-key property names (see CIM
1299         Requirements)>;
1300 }
1301 &smShowInstance ( $instance, #propertylist[] );
1302 &smEnd;

```

1303 **6.15 CIM\_PhysicalMemory**

1304 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1305 Table 15 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1306 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1307 target. Table 15 is for informational purposes only; in case of a conflict between Table 15 and  
 1308 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1309 information in Table 15.

1310 **Table 15 – Command Verb Requirements for CIM\_PhysicalMemory**

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 6.15.2.
Start	Not supported	
Stop	Not supported	

1311 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`, `load`,  
 1312 `reset`, `set`, `start`, and `stop`.

1313 **6.15.1 Ordering of Results**

1314 When results are returned for multiple instances of `CIM_ElementCapabilities`, implementations shall  
 1315 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1316 • Results for `CIM_PhysicalMemory` are unordered; therefore, no algorithm is defined.

1317 **6.15.2 Show**

1318 This section describes how to implement the `show` verb when applied to an instance of  
 1319 `CIM_PhysicalMemory`. Implementations shall support the use of the `show` verb with  
 1320 `CIM_PhysicalMemory`.

### 1321 **6.15.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container** 1322 **Instance**

1323 This command form is used to show many instances of CIM\_PhysicalMemory when  
1324 CIM\_PhysicalPackage is the container instance.

#### 1325 **6.15.2.1.1 Command Form**

```
1326 show <CIM_PhysicalMemory multiple instances>
```

#### 1327 **6.15.2.1.2 CIM Requirements**

1328 See CIM\_PhysicalMemory in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1329 mandatory properties.

#### 1330 **6.15.2.1.3 Behavior Requirements**

##### 1331 **6.15.2.1.3.1 Preconditions**

1332 In this section \$containerInstance represents the instance of CIM\_PhysicalPackage and is  
1333 associated to the targeted instances of CIM\_PhysicalMemory through the CIM\_Container association.

1334 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

##### 1335 **6.15.2.1.3.2 Pseudo Code**

```
1336 #propertylist[] = NULL;
1337 if ( false == #all) {
1338     #propertylist[] = <array of mandatory non-key property names (see CIM
1339         Requirements)>;
1340 }
1341 &smShowInstances ( "CIM_PhysicalMemory", "CIM_Container",
1342     $containerInstance.getInstancePath(), #propertylist[] );
1343 &smEnd;
```

### 1344 **6.15.2.2 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection** 1345 **Container Instance**

1346 This command form is used to show many instances of CIM\_PhysicalMemory when  
1347 CIM\_ConcreteCollection is the container instance.

#### 1348 **6.15.2.2.1 Command Form**

```
1349 show <CIM_PhysicalMemory multiple instances>
```

#### 1350 **6.15.2.2.2 CIM Requirements**

1351 See CIM\_PhysicalMemory in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1352 mandatory properties.

#### 1353 **6.15.2.2.3 Behavior Requirements**

##### 1354 **6.15.2.2.3.1 Preconditions**

1355 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection and is  
1356 associated to the targeted instances of CIM\_PhysicalMemory through the CIM\_MemberOfCollection  
1357 association.

1358 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1359 **6.15.2.2.3.2 Pseudo Code**

```

1360 #propertylist[] = NULL;
1361 if ( false == #all) {
1362     #propertylist[] = <array of mandatory non-key property names (see CIM
1363         Requirements)>;
1364 }
1365 &smShowInstances ( "CIM_PhysicalMemory", "CIM_MemberOfCollection",
1366     $containerInstance.getInstancePath(), #propertylist[] );
1367 &smEnd;

```

1368 **6.15.2.3 Show Command Form for a Single Instance Target**

1369 This command form is used to show a single instance of CIM\_PhysicalMemory.

1370 **6.15.2.3.1 Command Form**

```

1371 show <CIM_PhysicalMemory single instance>

```

1372 **6.15.2.3.2 CIM Requirements**

1373 See CIM\_PhysicalMemory in the "CIM Elements" section of the [Physical Asset Profile](#) for the list of  
1374 mandatory properties.

1375 **6.15.2.3.3 Behavior Requirements**1376 **6.15.2.3.3.1 Preconditions**

1377 In this section \$instance represents the targeted instance of CIM\_PhysicalMemory.

```

1378 $instance=<CIM_PhysicalMemory single instance>;

```

1379 #all is true if the "-all" option was specified with the command; otherwise, #all is false.

1380 **6.15.2.3.3.2 Pseudo Code**

```

1381 #propertylist[] = NULL;
1382 if ( false == #all) {
1383     #propertylist[] = <array of mandatory non-key property names (see CIM
1384         Requirements)>;
1385 }
1386 &smShowInstance ( $instance, #propertylist[] );
1387 &smEnd;

```

1388 **6.16 CIM\_PhysicalPackage**

1389 The cd, exit, help and version verbs shall be supported as described in [DSP0216](#).

1390 Table 16 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
1391 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and  
1392 target. Table 16 is for informational purposes only; in case of a conflict between Table 16 and  
1393 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
1394 information in Table 16.

1395

**Table 16 – Command Verb Requirements for CIM\_PhysicalPackage**

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 6.16.2.
Start	Not supported	
Stop	Not supported	

1396 No mapping is defined for the following verbs for the specified target: *create*, *delete*, *dump*, *exit*, *load*,  
 1397 *reset*, *set*, *start*, and *stop*.

1398 **6.16.1 Ordering of Results**

1399 When results are returned for multiple instances of *CIM\_ElementCapabilities*, implementations shall  
 1400 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1401 • Results for *CIM\_PhysicalPackage* are unordered; therefore, no algorithm is defined.

1402 **6.16.2 Show**

1403 This section describes how to implement the *show* verb when applied to an instance of  
 1404 *CIM\_PhysicalPackage*. Implementations shall support the use of the *show* verb with  
 1405 *CIM\_PhysicalPackage*.

1406 **6.16.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container  
 1407 Instance**

1408 This command form is used to show many instances of *CIM\_PhysicalPackage* when  
 1409 *CIM\_PhysicalPackage* is the container instance.

1410 **6.16.2.1.1 Command Form**

1411 `show <CIM_PhysicalPackage multiple instances>`

1412 **6.16.2.1.2 CIM Requirements**

1413 See *CIM\_PhysicalPackage* in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
 1414 mandatory properties.

1415 **6.16.2.1.3 Behavior Requirements**

1416 **6.16.2.1.3.1 Preconditions**

1417 In this section *\$containerInstance* represents the instance of *CIM\_PhysicalPackage* and is  
 1418 associated to the targeted instances of *CIM\_PhysicalPackage* through the *CIM\_Container* association.

1419 *#all* is true if the “-all” option was specified with the command; otherwise, *#all* is false.

1420 **6.16.2.1.3.2 Pseudo Code**

```

1421 #propertylist[] = NULL;
1422 if ( false == #all) {
1423     #propertylist[] = <array of mandatory non-key property names (see CIM
1424         Requirements)>;
1425 }
1426 &smShowInstances ( "CIM_PhysicalPackage", "CIM_Container",
1427     $containerInstance.getInstancePath(), #propertylist[] );
1428 &smEnd;

```

1429 **6.16.2.2 Show Command Form for Multiple Instances Target – CIM\_PhysicalConnector  
1430 Container Instance**

1431 This command form is used to show many instances of CIM\_PhysicalPackage when  
1432 CIM\_PhysicalConnector is the container instance.

1433 **6.16.2.2.1 Command Form**

```
1434 show <CIM_PhysicalPackage multiple instances>
```

1435 **6.16.2.2.2 CIM Requirements**

1436 See CIM\_PhysicalPackage in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1437 mandatory properties.

1438 **6.16.2.2.3 Behavior Requirements**1439 **6.16.2.2.3.1 Preconditions**

1440 In this section \$containerInstance represents the instance of CIM\_PhysicalConnector and is  
1441 associated to the targeted instances of CIM\_PhysicalPackage through the CIM\_PackageInConnector  
1442 association.

1443 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1444 **6.16.2.2.3.2 Pseudo Code**

```

1445 #propertylist[] = NULL;
1446 if ( false == #all) {
1447     #propertylist[] = <array of mandatory non-key property names (see CIM
1448         Requirements)>;
1449 }
1450 &smShowInstances ( "CIM_PhysicalPackage", "CIM_PackageInConnector",
1451     $containerInstance.getInstancePath(), #propertylist[] );
1452 &smEnd;

```

1453 **6.16.2.3 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection  
1454 Container Instance**

1455 This command form is used to show many instances of CIM\_PhysicalPackage when  
1456 CIM\_ConcreteCollection is the container instance.

1457 **6.16.2.3.1 Command Form**

```
1458 show <CIM_PhysicalPackage multiple instances>
```



### 1459 6.16.2.3.2 CIM Requirements

1460 See CIM\_PhysicalPackage in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1461 mandatory properties.

### 1462 6.16.2.3.3 Behavior Requirements

#### 1463 6.16.2.3.3.1 Preconditions

1464 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection and is  
1465 associated to the targeted instances of CIM\_PhysicalPackage through the CIM\_MemberOfCollection  
1466 association.

1467 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

#### 1468 6.16.2.3.3.2 Pseudo Code

```
1469 #propertylist[] = NULL;
1470 if ( false == #all) {
1471     #propertylist[] = <array of mandatory non-key property names (see CIM
1472         Requirements)>;
1473 }
1474 &smShowInstances ( "CIM_PhysicalPackage", "CIM_MemberOfCollection",
1475     $containerInstance.getInstancePath(), #propertylist[] );
1476 &smEnd;
```

### 1477 6.16.2.4 Show Command Form for a Single Instance Target

1478 This command form is used to show a single instance of CIM\_PhysicalPackage.

#### 1479 6.16.2.4.1 Command Form

```
1480 show <CIM_PhysicalPackage single instance>
```

#### 1481 6.16.2.4.2 CIM Requirements

1482 See CIM\_PhysicalPackage in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1483 mandatory properties.

#### 1484 6.16.2.4.3 Behavior Requirements

##### 1485 6.16.2.4.3.1 Preconditions

1486 In this section \$instance represents the targeted instance of CIM\_PhysicalPackage.

```
1487 $instance=<CIM_PhysicalPackage single instance>;
```

1488 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

##### 1489 6.16.2.4.3.2 Pseudo Code

```
1490 #propertylist[] = NULL;
1491 if ( false == #all) {
1492     #propertylist[] = <array of mandatory non-key property names (see CIM
1493         Requirements)>;
1494 }
1495 &smShowInstance ( $instance, #propertylist[] );
1496 &smEnd;
```

1497 **6.17 CIM\_Rack**1498 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1499 Table 17 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1500 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1501 target. Table 17 is for informational purposes only; in case of a conflict between Table 17 and  
 1502 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1503 information in Table 17.

1504 **Table 17 – Command Verb Requirements for CIM\_Rack**

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 6.17.2.
Start	Not supported	
Stop	Not supported	

1505 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,  
 1506 `load`, `reset`, `set`, `start`, and `stop`.

1507 **6.17.1 Ordering of Results**

1508 When results are returned for multiple instances of `CIM_ElementCapabilities`, implementations shall  
 1509 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1510 • Results for `CIM_Rack` are unordered; therefore, no algorithm is defined.

1511 **6.17.2 Show**

1512 This section describes how to implement the `show` verb when applied to an instance of `CIM_Rack`.  
 1513 Implementations shall support the use of the `show` verb with `CIM_Rack`.

1514 **6.17.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container Instance**

1516 This command form is used to show many instances of `CIM_Rack` when `CIM_PhysicalPackage` is the  
 1517 container instance.

1518 **6.17.2.1.1 Command Form**

1519 `show <CIM_Rack multiple instances>`

1520 **6.17.2.1.2 CIM Requirements**

1521 See `CIM_Rack` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 1522 properties.

### 1523 6.17.2.1.3 Behavior Requirements

#### 1524 6.17.2.1.3.1 Preconditions

1525 In this section `$containerInstance` represents the instance of `CIM_PhysicalPackage` and is  
1526 associated to the targeted instances of `CIM_Rack` through the `CIM_Container` association.

1527 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

#### 1528 6.17.2.1.3.2 Pseudo Code

```
1529 #propertylist[] = NULL;
1530 if ( false == #all) {
1531     #propertylist[] = <array of mandatory non-key property names (see CIM
1532         Requirements)>;
1533 }
1534 &smShowInstances ( "CIM_Rack", "CIM_Container", $containerInstance.getInstancePath(),
1535     #propertylist[] );
1536 &smEnd;
```

### 1537 6.17.2.2 Show Command Form for Multiple Instances Target – CIM\_PhysicalConnector 1538 Container Instance

1539 This command form is used to show many instances of `CIM_Rack` when `CIM_PhysicalConnector` is the  
1540 container instance.

#### 1541 6.17.2.2.1 Command Form

```
1542 show <CIM_Rack multiple instances>
```

#### 1543 6.17.2.2.2 CIM Requirements

1544 See `CIM_Rack` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
1545 properties.

#### 1546 6.17.2.2.3 Behavior Requirements

##### 1547 6.17.2.2.3.1 Preconditions

1548 In this section `$containerInstance` represents the instance of `CIM_PhysicalConnector` and is  
1549 associated to the targeted instances of `CIM_Rack` through the `CIM_PackageInConnector` association.

1550 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

#### 1551 6.17.2.2.3.2 Pseudo Code

```
1552 #propertylist[] = NULL;
1553 if ( false == #all) {
1554     #propertylist[] = <array of mandatory non-key property names (see CIM
1555         Requirements)>;
1556 }
1557 &smShowInstances ( "CIM_Rack", "CIM_PackageInConnector",
1558     $containerInstance.getInstancePath(), #propertylist[] );
1559 &smEnd;
```

1560 **6.17.2.3 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection**  
 1561 **Container Instance**

1562 This command form is used to show many instances of CIM\_Rack when CIM\_ConcreteCollection is the  
 1563 container instance.

1564 **6.17.2.3.1 Command Form**

```
1565 show <CIM_Rack multiple instances>
```

1566 **6.17.2.3.2 CIM Requirements**

1567 See CIM\_Rack in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 1568 properties.

1569 **6.17.2.3.3 Behavior Requirements**

1570 **6.17.2.3.3.1 Preconditions**

1571 In this section \$containerInstance represents the instance of CIM\_ConcreteCollection and is  
 1572 associated to the targeted instances of CIM\_Rack through the CIM\_MemberOfCollection association.

1573 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1574 **6.17.2.3.3.2 Pseudo Code**

```
1575 #propertylist[] = NULL;
1576 if ( false == #all) {
1577     #propertylist[] = <array of mandatory non-key property names (see CIM
1578         Requirements)>;
1579 }
1580 &smShowInstances ( "CIM_Rack", "CIM_MemberOfCollection",
1581     $containerInstance.getInstancePath(), #propertylist[] );
1582 &smEnd;
```

1583 **6.17.2.4 Show Command Form for a Single Instance Target**

1584 This command form is used to show a single instance of CIM\_Rack.

1585 **6.17.2.4.1 Command Form**

```
1586 show <CIM_Rack single instance>
```

1587 **6.17.2.4.2 CIM Requirements**

1588 See CIM\_Rack in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 1589 properties.

1590 **6.17.2.4.3 Behavior Requirements**

1591 **6.17.2.4.3.1 Preconditions**

1592 In this section \$instance represents the targeted instance of CIM\_Rack.

```
1593 $instance=<CIM_Rack single instance>;
```

1594 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1595 **6.17.2.4.3.2 Pseudo Code**

```

1596 #propertylist[] = NULL;
1597 if ( false == #all) {
1598     #propertylist[] = <array of mandatory non-key property names (see CIM
1599         Requirements)>;
1600 }
1601 &smShowInstance ( $instance, #propertylist[] );
1602 &smEnd;
    
```

1603 **6.18 CIM\_Realizes**

1604 The cd, exit, help, and version verbs shall be supported as described in [DSP0216](#).

1605 Table 18 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1606 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1607 target. Table 18 is for informational purposes only; in case of a conflict between Table 18 and  
 1608 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1609 information in Table 18.

1610 **Table 18 – Command Verb Requirements for CIM\_Realizes**

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 6.18.2.
Start	Not supported	
Stop	Not supported	

1611 No mapping is defined for the following verbs for the specified target: create, delete, dump, exit,  
 1612 load, reset, set, start, and stop.

1613 **6.18.1 Ordering of Results**

1614 When results are returned for multiple instances of CIM\_ElementCapabilities, implementations shall  
 1615 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1616 • Results for CIM\_Realizes are unordered; therefore, no algorithm is defined.

1617 **6.18.2 Show**

1618 This section describes how to implement the show verb when applied to an instance of CIM\_Realizes.  
 1619 Implementations shall support the use of the show verb with CIM\_Realizes.

1620 **6.18.2.1 Show Command Form for Multiple Instances Target – Subclass CIM\_LogicalDevice**  
1621 **Reference**

1622 This command form is used to show many instances of CIM\_Realizes. This command form corresponds  
1623 to a `show` command issued against instances of CIM\_Realizes where only one reference is specified and  
1624 the reference is the instance of a subclass of CIM\_LogicalDevice.

1625 **6.18.2.1.1 Command Form**

```
1626 show <CIM_Realizes multiple instances>
```

1627 **6.18.2.1.2 CIM Requirements**

1628 See CIM\_Realizes in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
1629 properties and CIM classes that can be referenced.

1630 **6.18.2.1.3 Behavior Requirements**

1631 **6.18.2.1.3.1 Preconditions**

1632 In this section `$instance` represents the instance of a subclass of CIM\_LogicalDevice which is  
1633 referenced by CIM\_Realizes.

1634 **6.18.2.1.3.2 Pseudo Code**

```
1635 &smShowAssociationInstances ( "CIM_Realizes", $instance.getInstancePath() );  
1636 &smEnd;
```

1637 **6.18.2.2 Show Command Form for Multiple Instances Target – Subclass CIM\_PhysicalElement**  
1638 **Reference**

1639 This command form is used to show many instances of CIM\_Realizes. This command form corresponds  
1640 to a `show` command issued against instances of CIM\_Realizes where only one reference is specified and  
1641 the reference is the instance of a subclass of CIM\_PhysicalElement.

1642 **6.18.2.2.1 Command Form**

```
1643 show <CIM_Realizes multiple instances>
```

1644 **6.18.2.2.2 CIM Requirements**

1645 See CIM\_Realizes in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
1646 properties and CIM classes that can be referenced.

1647 **6.18.2.2.3 Behavior Requirements**

1648 **6.18.2.2.3.1 Preconditions**

1649 In this section `$instance` represents the instance of a subclass of CIM\_PhysicalElement which is  
1650 referenced by CIM\_Realizes.

1651 **6.18.2.2.3.2 Pseudo Code**

```
1652 &smShowAssociationInstances ( "CIM_Realizes", $instance.getInstancePath() );  
1653 &smEnd;
```

1654 **6.18.2.3 Show Command Form for a Single Instance – Both References**

1655 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
 1656 a `show` command issued against `CIM_Realizes` where both references are specified and therefore the  
 1657 desired instance is unambiguously identified.

1658 **6.18.2.3.1 Command Form**

```
show <CIM_Realizes single instance>
```

1660 **6.18.2.3.2 CIM Requirements**

1661 See `CIM_Realizes` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
 1662 properties and CIM classes that can be referenced.

1663 **6.18.2.3.3 Behavior Requirements**

1664 **6.18.2.3.3.1 Preconditions**

1665 In this section `$instanceA` represents the referenced instance of a subclass of `CIM_LogicalDevice`  
 1666 through `CIM_Realizes` association. `$instanceB` represents the instance of a subclass of  
 1667 `CIM_PhysicalElement` which is referenced by `CIM_Realizes`.

1668 **6.18.2.3.3.2 Pseudo Code**

```
&smShowAssociationInstance ( "CIM_Realizes", $instanceA.getInstancePath(),  

    $instanceB.getInstancePath() );  

&smEnd;
```

1672 **6.19 CIM\_Slot**

1673 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1674 Table 19 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1675 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1676 target. Table 19 is for informational purposes only; in case of a conflict between Table 19 and  
 1677 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1678 information in Table 19.

1679 **Table 19 – Command Verb Requirements for CIM\_Slot**

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 6.19.2.
Start	Not supported	
Stop	Not supported	

1680 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,  
 1681 `load`, `reset`, `set`, `start`, and `stop`.

### 1682 6.19.1 Ordering of Results

1683 When results are returned for multiple instances of CIM\_ElementCapabilities, implementations shall  
1684 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1685 • Results for CIM\_Slot are unordered; therefore, no algorithm is defined.

### 1686 6.19.2 Show

1687 This section describes how to implement the `show` verb when applied to an instance of CIM\_Slot.  
1688 Implementations shall support the use of the `show` verb with CIM\_Slot.

#### 1689 6.19.2.1 Show Command Form for Multiple Instances Target – CIM\_PhysicalPackage Container 1690 Instance

1691 This command form is used to show many instances of CIM\_Slot when CIM\_PhysicalPackage is the  
1692 container instance.

##### 1693 6.19.2.1.1 Command Form

```
1694 show <CIM_Slot multiple instances>
```

##### 1695 6.19.2.1.2 CIM Requirements

1696 See CIM\_Slot in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
1697 properties.

##### 1698 6.19.2.1.3 Behavior Requirements

###### 1699 6.19.2.1.3.1 Preconditions

1700 In this section `$containerInstance` represents the instance of CIM\_PhysicalPackage and is  
1701 associated to the targeted instances of CIM\_Slot through the CIM\_Container association.

1702 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

###### 1703 6.19.2.1.3.2 Pseudo Code

```
1704 #propertylist[] = NULL;
1705 if ( false == #all) {
1706     #propertylist[] = <array of mandatory non-key property names (see CIM
1707         Requirements)>;
1708 }
1709 &smShowInstances ( "CIM_Slot", "CIM_Container", $containerInstance.getInstancePath(),
1710     #propertylist[] );
1711 &smEnd;
```

#### 1712 6.19.2.2 Show Command Form for Multiple Instances Target – CIM\_ConcreteCollection 1713 Container Instance

1714 This command form is used to show many instances of CIM\_Slot when CIM\_ConcreteCollection is the  
1715 container instance.

##### 1716 6.19.2.2.1 Command Form

```
1717 show <CIM_Slot multiple instances>
```



### 1718 6.19.2.2.2 CIM Requirements

1719 See CIM\_Slot in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
1720 properties.

### 1721 6.19.2.2.3 Behavior Requirements

#### 1722 6.19.2.2.3.1 Preconditions

1723 In this section `$containerInstance` represents the instance of `CIM_ConcreteCollection` and is  
1724 associated to the targeted instances of `CIM_Slot` through the `CIM_MemberOfCollection` association.

1725 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

#### 1726 6.19.2.2.3.2 Pseudo Code

```
1727 #propertylist[] = NULL;
1728 if ( false == #all) {
1729     #propertylist[] = <array of mandatory non-key property names (see CIM
1730         Requirements)>;
1731 }
1732 &smShowInstances ( "CIM_Slot", "CIM_MemberOfCollection",
1733     $containerInstance.getInstancePath(), #propertylist[] );
1734 &smEnd;
```

### 1735 6.19.2.3 Show Command Form for a Single Instance Target

1736 This command form is used to show a single instance of `CIM_Slot`.

#### 1737 6.19.2.3.1 Command Form

```
1738 show <CIM_Slot single instance>
```

#### 1739 6.19.2.3.1.1 CIM Requirements

1740 See `CIM_Slot` in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of mandatory  
1741 properties.

#### 1742 6.19.2.3.2 Behavior Requirements

##### 1743 6.19.2.3.2.1 Preconditions

1744 In this section `$instance` represents the targeted instance of `CIM_Slot`.

```
1745 $instance=<CIM_Slot single instance>;
```

1746 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

##### 1747 6.19.2.3.2.2 Pseudo Code

```
1748 #propertylist[] = NULL;
1749 if ( false == #all) {
1750     #propertylist[] = <array of mandatory non-key property names (see CIM
1751         Requirements)>;
1752 }
1753 &smShowInstance ( $instance, #propertylist[] );
1754 &smEnd;
```

1755 **6.20 CIM\_SystemPackaging**1756 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1757 Table 20 lists each SM CLP verb, the required level of support for the verb in conjunction with the target  
 1758 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and  
 1759 target. Table 20 is for informational purposes only; in case of a conflict between Table 20 and  
 1760 requirements detailed in the following sections, the text detailed in the following sections supersedes the  
 1761 information in Table 20.

1762 **Table 20 – Command Verb Requirements for CIM\_SystemPackaging**

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 6.20.2.
Start	Not supported	
Stop	Not supported	

1763 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `exit`,  
 1764 `load`, `reset`, `set`, `start`, and `stop`.

1765 **6.20.1 Ordering of Results**

1766 When results are returned for multiple instances of `CIM_ElementCapabilities`, implementations shall  
 1767 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1768 • Results for `CIM_SystemPackaging` are unordered; therefore, no algorithm is defined.

1769 **6.20.2 Show**

1770 This section describes how to implement the `show` verb when applied to an instance of  
 1771 `CIM_SystemPackaging`. Implementations shall support the use of the `show` verb with  
 1772 `CIM_SystemPackaging`.

1773 **6.20.2.1 Show Command Form for Multiple Instances Target – CIM\_System**

1774 This command form is used to show many instances of `CIM_SystemPackaging`. This command form  
 1775 corresponds to a `show` command issued against `CIM_SystemPackaging` where only one reference is  
 1776 specified and the reference is to an instance of `CIM_System`.

1777 **6.20.2.1.1 Command Form**

1778 `show <CIM_SystemPackaging multiple instances>`

### 1779 6.20.2.1.2 CIM Requirements

1780 See CIM\_SystemPackaging in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1781 mandatory properties and CIM classes that can be referenced.

### 1782 6.20.2.1.3 Behavior Requirements

#### 1783 6.20.2.1.3.1 Preconditions

1784 In this section `$instance` represents the instance of CIM\_System which is referenced by  
1785 CIM\_SystemPackaging.

#### 1786 6.20.2.1.3.2 Pseudo Code

```
1787 &smShowAssociationInstances ( "CIM_SystemPackaging", $instance.getInstancePath() );  
1788 &smEnd;
```

### 1789 6.20.2.2 Show Command Form for Multiple Instances – CIM\_PhysicalPackage Reference

1790 This command form is used to show multiple instances of CIM\_SystemPackaging. This command form  
1791 corresponds to a `show` command issued against multiple instances of CIM\_SystemPackaging where only  
1792 one reference is specified and the reference is to an instance of CIM\_PhysicalPackage or a subclass of  
1793 CIM\_PhysicalPackage.

#### 1794 6.20.2.2.1 Command Form

```
1795 show <CIM_SystemPackaging multiple instances>
```

#### 1796 6.20.2.2.2 CIM Requirements

1797 See CIM\_SystemPackaging in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1798 mandatory properties and CIM classes that can be referenced.

#### 1799 6.20.2.2.3 Behavior Requirements

##### 1800 6.20.2.2.3.1 Preconditions

1801 In this section `$instance` represents the instance of CIM\_PhysicalPackage or a subclass of  
1802 CIM\_PhysicalPackage which is referenced by CIM\_SystemPackaging.

##### 1803 6.20.2.2.3.2 Pseudo Code

```
1804 &smShowAssociationInstances ( "CIM_SystemPackaging", $instance.getInstancePath() );  
1805 &smEnd;
```

### 1806 6.20.2.3 Show Command Form for a Single Instance – Both References

1807 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
1808 a `show` command issued against CIM\_SystemPackaging where both references are specified and  
1809 therefore the desired instance is unambiguously identified.

#### 1810 6.20.2.3.1 Command Form

```
1811 show <CIM_SystemPackaging single instance>
```

#### 1812 6.20.2.3.2 CIM Requirements

1813 See CIM\_SystemPackaging in the “CIM Elements” section of the [Physical Asset Profile](#) for the list of  
1814 mandatory properties and CIM classes that can be referenced.

1815 **6.20.2.3.3 Behavior Requirements**1816 **6.20.2.3.3.1 Preconditions**

1817 In this section `$instanceA` represents the referenced instance of `CIM_System` through  
1818 `CIM_SystemPackaging` association. `$instanceB` represents the instance of `CIM_PhysicalPackage` or a  
1819 subclass of `CIM_PhysicalPackage` which is referenced by `CIM_SystemPackaging`.

1820 **6.20.2.3.3.2 Pseudo Code**

```
1821 &smShowAssociationInstance ( "CIM_SystemPackaging", $instanceA.getInstancePath(),  
1822     $instanceB.getInstancePath() );  
1823 &smEnd;
```

1824

**ANNEX A**  
(informative)

**Change Log**

1825  
1826  
1827  
1828  
1829

Version	Date	Author	Description
1.0.0	2009-07-14		DMTF Standard Release

1830