



1
2
3
4

Document Number: DSP0814

Date: 2009-07-14

Version: 1.0.0

5 **Fan Profile SM CLP Command 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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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_AssociatedCooling	9
45	6.2 CIM_ElementCapabilities	12
46	6.3 CIM_EnabledLogicalElementCapabilities.....	14
47	6.4 CIM_IsSpare	16
48	6.5 CIM_MemberOfCollection	19
49	6.6 CIM_Fan	21
50	6.7 CIM_RedundancySet.....	28
51	6.8 CIM_SystemDevice	34
52	6.9 CIM_OwningCollectionElement.....	36
53	ANNEX A (informative) Change Log.....	39
54		

55 Tables

56	Table 1 – Command Verb Requirements for CIM_AssociatedCooling	10
57	Table 2 – Command Verb Requirements for CIM_ElementCapabilities	12
58	Table 3 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities.....	14
59	Table 4 – Command Verb Requirements for CIM_IsSpare	16
60	Table 5 – Command Verb Requirements for CIM_MemberOfCollection	19
61	Table 6 – Command Verb Requirements for CIM_Fan	21
62	Table 7 – Command Verb Requirements for CIM_RedundancySet.....	28
63	Table 8 – Command Verb Requirements for CIM_SystemDevice	34
64	Table 9 – Command Verb Requirements for CIM_OwningCollectionElement.....	36
65		

67

Foreword

68 The *Fan Profile SM CLP Command Mapping Specification* (DSP0814) was prepared by the Server
69 Management Working Group.

70 **Conventions**

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

73 **Acknowledgements**

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

- 76 • Khachatur Papanyan – Dell
- 77 • Jon Hass – Dell
- 78 • Enoch Suen – Dell
- 79 • Jeff Hilland – HP
- 80 • Christina Shaw – HP
- 81 • Aaron Merkin – IBM
- 82 • Perry Vincent – Intel
- 83 • John Leung – Intel
- 84 • John Ackerley – Sun Microsystems

85

86

Introduction

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

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

92 Fan Profile SM CLP Command Mapping Specification

93 1 Scope

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

96 2 Normative References

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

100 2.1 Approved References

101 DMTF DSP0216, *SM CLP-to-CIM Common Mapping Specification 1.0*,
102 http://www.dmtf.org/standards/published_documents/DSP0216_1.0.pdf

103 DMTF DSP1013, *Fan Profile 1.0*,
104 http://www.dmtf.org/standards/published_documents/DSP1013_1.0.pdf

105 SNIA, *Storage Management Initiative Specification (SMI-S) 1.1.0*,
106 http://www.snia.org/tech_activities/standards/curr_standards/smi

107 2.2 Other References

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

110 3 Terms and Definitions

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

112 3.1

113 **can**

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

115 3.2

116 **cannot**

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

118 3.3

119 **conditional**

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

122 3.4

123 **mandatory**

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

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

150 **4 Symbols and Abbreviated Terms**

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

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

- 164 **4.5**
 165 **SM**
 166 Server Management
- 167 **4.6**
 168 **SMI-S**
 169 Storage Management Initiative Specification
- 170 **4.7**
 171 **SNIA**
 172 Storage Networking Industry Association
- 173 **4.8**
 174 **UFsT**
 175 User Friendly selection Tag

176 **5 Recipes**

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

- 179 • smResetRSC
- 180 • smShowInstance
- 181 • smShowInstances
- 182 • smShowAssociationInstance
- 183 • smShowAssociationInstances
- 184 • smStartRSC
- 185 • smStopRSC

186 **6 Mappings**

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

190 **6.1 CIM_AssociatedCooling**

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

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

197

Table 1 – Command Verb Requirements for CIM_AssociatedCooling

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	

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

200 6.1.1 Ordering of Results

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

- 203 • Results for `CIM_AssociatedCooling` are unordered; therefore, no algorithm is defined.

204 6.1.2 Show

205 This section describes how to implement the `show` verb when applied to an instance of
 206 `CIM_AssociatedCooling`. Implementations shall support the use of the `show` verb with
 207 `CIM_AssociatedCooling`.

208 6.1.2.1 Show Command Form for Multiple Instances Target – CIM_ManagedSystemElement 209 Reference

210 This command form is used to show many instances of `CIM_AssociatedCooling`. This command form
 211 corresponds to a `show` command issued against the instance of `CIM_AssociatedCooling` where only one
 212 reference is specified and the reference is to the instance of `CIM_ManagedSystemElement`.

213 6.1.2.1.1 Command Form

```
214 show <CIM_AssociatedCooling multiple instances>
```

215 6.1.2.1.2 CIM Requirements

216 See `CIM_AssociatedCooling` in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
 217 properties.

218 6.1.2.1.3 Behavior Requirements

219 6.1.2.1.3.1 Preconditions

220 `$instance` represents the instance of a `CIM_ManagedSystemElement`, which is referenced by
 221 `CIM_AssociatedCooling`.

222 6.1.2.1.3.2 Pseudo Code

```
223 &smShowAssociationInstances ( "CIM_AssociatedCooling", $instance.getObjectPath() );
224 &smEnd;
```

225 6.1.2.2 Show Command Form for Multiple Instance Target – CIM_Fan Reference

226 This command form is used to show multiple instances of CIM_AssociatedCooling. This command form
 227 corresponds to a `show` command issued against multiple instances of CIM_AssociatedCooling, where
 228 only one reference is specified and the reference is to the instance of CIM_Fan.

229 6.1.2.2.1 Command Form

```
230 show <CIM_AssociatedCooling multiple instances>
```

231 6.1.2.2.2 CIM Requirements

232 See CIM_AssociatedCooling in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
 233 properties.

234 6.1.2.2.3 Behavior Requirements

235 6.1.2.2.3.1 Preconditions

236 `$instance` represents the instance of CIM_Fan which is referenced by CIM_AssociatedCooling.

237 6.1.2.2.3.2 Pseudo Code

```
238 &smShowAssociationInstances ( "CIM_AssociatedCooling", $instance.getObjectPath() );
239 &smEnd;
```

240 6.1.2.3 Show Command Form for a Single Instance Target – Both References

241 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 242 a `show` command issued against CIM_AssociatedCooling where both references are specified and
 243 therefore the desired instance is unambiguously identified.

244 6.1.2.3.1 Command Form

```
245 show <CIM_AssociatedCooling single instance>
```

246 6.1.2.3.2 CIM Requirements

247 See CIM_AssociatedCooling in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
 248 properties.

249 6.1.2.3.3 Behavior Requirements

250 6.1.2.3.3.1 Preconditions

251 `$instanceA` represents the referenced instance of CIM_Fan through CIM_AssociatedCooling
 252 association.

253 `$instanceB` represents the instance of CIM_ManagedSystemElement which is referenced by
 254 CIM_AssociatedCooling.

255 6.1.2.3.3.2 Pseudo Code

```
256 &smShowAssociationInstance ( "CIM_AssociatedCooling", $instanceA.getObjectPath(),
257     $instanceB.getObjectPath() );
258 &smEnd;
```

259 6.2 CIM_ElementCapabilities

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

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

266 **Table 2 – 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 6.2.2.
Start	Not supported	
Stop	Not supported	

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

269 6.2.1 Ordering of Results

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

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

273 6.2.2 Show

274 This section describes how to implement the `show` verb when applied to an instance of
 275 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with
 276 `CIM_ElementCapabilities`.

277 6.2.2.1 Show Command Form for Multiple Instances Target – 278 CIM_EnabledLogicalElementCapabilities Reference

279 This command form is used to show many instances of `CIM_ElementCapabilities`. This command form
 280 corresponds to a `show` command issued against instances of `CIM_ElementCapabilities` where only one
 281 reference is specified and the reference is to an instance of `CIM_EnabledLogicalElementCapabilities`.

282 6.2.2.1.1 Command Form

283 `show <CIM_ElementCapabilities multiple instances>`

284 6.2.2.1.2 CIM Requirements

285 See CIM_ElementCapabilities in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
286 properties.

287 6.2.2.1.3 Behavior Requirements

288 6.2.2.1.3.1 Preconditions

289 \$instance represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by
290 CIM_ElementCapabilities.

291 6.2.2.1.3.2 Pseudo Code

```
292 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
293 &smEnd;
```

294 6.2.2.2 Show Command Form for a Single Instance – CIM_Fan Reference

295 This command form is used to show a single instance of CIM_ElementCapabilities. This command form
296 corresponds to a show command issued against a single instance of CIM_ElementCapabilities where
297 only one reference is specified and the reference is to the instance of CIM_Fan.

298 6.2.2.2.1 Command Form

```
299 show <CIM_ElementCapabilities single instance>
```

300 6.2.2.2.2 CIM Requirements

301 See CIM_ElementCapabilities in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
302 properties.

303 6.2.2.2.3 Behavior Requirements

304 6.2.2.2.3.1 Preconditions

305 \$instance represents the instance of CIM_Fan which is referenced by CIM_ElementCapabilities.

306 6.2.2.2.3.2 Pseudo Code

```
307 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
308 &smEnd;
```

309 6.2.2.3 Show Command Form for a Single Instance Target – Both References

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

313 6.2.2.3.1 Command Form

```
314 show <CIM_ElementCapabilities single instance>
```

315 6.2.2.3.2 CIM Requirements

316 See CIM_ElementCapabilities in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
317 properties.

318 **6.2.2.3.3 Behavior Requirements**319 **6.2.2.3.3.1 Preconditions**

320 \$instanceA represents the referenced instance of CIM_Fan through CIM_ElementCapabilities
321 association.

322 \$instanceB represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by
323 CIM_ElementCapabilities.

324 **6.2.2.3.3.2 Pseudo Code**

```
325 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
326     $instanceB.getObjectPath() );
327 &smEnd;
```

328 **6.3 CIM_EnabledLogicalElementCapabilities**

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

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

335 **Table 3 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities**

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	

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

338 **6.3.1 Ordering of Results**

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

- 341 • Results for CIM_EnabledLogicalElementCapabilities are unordered: therefore, no algorithm is
342 defined.

343 **6.3.2 Show**

344 This section describes how to implement the `show` verb when applied to an instance of
 345 `CIM_EnabledLogicalElementCapabilities`. Implementations shall support the use of the `show` verb with
 346 `CIM_EnabledLogicalElementCapabilities`.

347 **6.3.2.1 Show Command Form for Multiple Instances Target**

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

349 **6.3.2.1.1 Command Form**

```
350 show <CIM_EnabledLogicalElementCapabilities multiple instances>
```

351 **6.3.2.1.2 CIM Requirements**

352 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Fan Profile](#) for the list
 353 of mandatory properties.

354 **6.3.2.1.3 Behavior Requirements**

355 **6.3.2.1.3.1 Preconditions**

356 `$containerInstance` represents the instance of `CIM_ConcreteCollection` with `ElementName` property
 357 that contains “Capabilities” and is associated to the targeted instances of
 358 `CIM_EnabledLogicalElementCapabilities` through the `CIM_MemberOfCollection` association.

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

360 **6.3.2.1.3.2 Pseudo Code**

```
361 #propertylist[] = NULL;
362 if ( false == #all)
363     {
364         #propertylist[] = <array of mandatory non-key property names (see CIM
365             Requirements)>;
366     }
367 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
368     $containerInstance.getObjectPath(), #propertylist[] );
369 &smEnd;
```

370 **6.3.2.2 Show Command Form for a Single Instance Target**

371 This command form is used to show a single instance of `CIM_EnabledLogicalElementCapabilities`.

372 **6.3.2.2.1 Command Form**

```
373 show <CIM_EnabledLogicalElementCapabilities single instance>
```

374 **6.3.2.2.2 CIM Requirements**

375 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Fan Profile](#) for the list
 376 of mandatory properties.

377 **6.3.2.2.3 Behavior Requirements**378 **6.3.2.2.3.1 Preconditions**

379 \$instance represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

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

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

382 **6.3.2.2.3.2 Pseudo Code**

```

383 #propertylist[] = NULL;
384 if ( false == #all)
385 {
386     #propertylist[] = <array of mandatory non-key property names (see CIM
387         Requirements)>;
388 }
389 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
390 &smEnd;

```

391 **6.4 CIM_IsSpare**392 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

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

398 **Table 4 – Command Verb Requirements for CIM_IsSpare**

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	

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

401 **6.4.1 Ordering of Results**

402 When results are returned for multiple instances of CIM_IsSpare, implementations shall utilize the
 403 following algorithm to produce the natural (that is, default) ordering:

- 404 • Results for CIM_IsSpare are unordered; therefore, no algorithm is defined.

405 6.4.2 Show

406 This section describes how to implement the `show` verb when applied to an instance of `CIM_IsSpare`.
 407 Implementations shall support the use of the `show` verb with `CIM_IsSpare`.

408 6.4.2.1 Show Command Form for Multiple Instances Target – `CIM_RedundancySet` Reference

409 This command form is used to show many instances of `CIM_IsSpare`. This command form corresponds to
 410 a `show` command issued against instances of `CIM_IsSpare` where only one reference is specified and the
 411 reference is to an instance of `CIM_RedundancySet`.

412 6.4.2.1.1 Command Form

```
413 show <CIM_IsSpare multiple instances>
```

414 6.4.2.1.2 CIM Requirements

415 See `CIM_IsSpare` in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory properties.

416 6.4.2.1.3 Behavior Requirements

417 6.4.2.1.3.1 Preconditions

418 `$instance` represents the instance of `CIM_RedundancySet` which is referenced by `CIM_IsSpare`.

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

420 6.4.2.1.3.2 Pseudo Code

```
421 #propertylist[] = NULL;
422 if ( false == #all)
423 {
424     #propertylist[] = <array of mandatory non-key property names (see CIM
425     Requirements)>;
426 }
427 &smShowAssociationInstances ( "CIM_IsSpare", $instance.getObjectPath(),
428     #propertylist[] );
429 &smEnd;
```

430 6.4.2.2 Show Command Form for a Single Instance – `CIM_Fan` Reference

431 This command form is used to show a single instance of `CIM_IsSpare`. This command form corresponds
 432 to a `show` command issued against a single instance of `CIM_IsSpare` where only one reference is
 433 specified and the reference is to the instance of `CIM_Fan`.

434 6.4.2.2.1 Command Form

```
435 show <CIM_IsSpare single instance>
```

436 6.4.2.2.2 CIM Requirements

437 See `CIM_IsSpare` in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory properties.

438 **6.4.2.2.3 Behavior Requirements**439 **6.4.2.2.3.1 Preconditions**

440 \$instance represents the instance of CIM_Fan which is referenced by CIM_IsSpare.

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

442 **6.4.2.2.3.2 Pseudo Code**

```
443 #propertylist[] = NULL;
444 if ( false == #all)
445     {
446         #propertylist[] = <array of mandatory non-key property names (see CIM
447             Requirements)>;
448     }
449 &smShowAssociationInstances ( "CIM_IsSpare", $instance.getObjectPath(),
450     #propertylist[]);
451 &smEnd;
```

452 **6.4.2.3 Show Command Form for a Single Instance Target – Both References**

453 This command form is for the *show* verb applied to a single instance. This command form corresponds to
 454 a *show* command issued against CIM_IsSpare where both references are specified and therefore the
 455 desired instance is unambiguously identified.

456 **6.4.2.3.1 Command Form**

```
457 show <CIM_IsSpare single instance>
```

458 **6.4.2.3.2 CIM Requirements**

459 See CIM_IsSpare in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory properties.

460 **6.4.2.3.3 Behavior Requirements**461 **6.4.2.3.3.1 Preconditions**

462 \$instanceA represents the referenced instance of CIM_Fan through CIM_IsSpare association.

463 \$instanceB represents the instance of CIM_RedundancySet which is referenced by CIM_IsSpare.

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

465 **6.4.2.3.3.2 Pseudo Code**

```
466 #propertylist[] = NULL;
467 if ( false == #all)
468     {
469         #propertylist[] = <array of mandatory non-key property names (see CIM
470             Requirements)>;
471     }
472 &smShowAssociationInstance ( "CIM_IsSpare", $instanceA.getObjectPath(),
473     $instanceB.getObjectPath(), #propertylist[] );
474 &smEnd;
```

475 **6.5 CIM_MemberOfCollection**

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

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

482 **Table 5 – Command Verb Requirements for CIM_MemberOfCollection**

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	

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

485 **6.5.1 Ordering of Results**

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

- 488 • Results for `CIM_MemberOfCollection` are unordered; therefore, no algorithm is defined.

489 **6.5.2 Show**

490 This section describes how to implement the `show` verb when applied to an instance of
 491 `CIM_MemberOfCollection`. Implementations shall support the use of the `show` verb with
 492 `CIM_MemberOfCollection`.

493 **6.5.2.1 Show Command Form for Multiple Instances Target – CIM_RedundancySet Reference**

494 This command form is used to show many instances of `CIM_MemberOfCollection`. This command form
 495 corresponds to a `show` command issued against instances of `CIM_MemberOfCollection` where only one
 496 reference is specified and the reference is to the instance of `CIM_RedundancySet`.

497 **6.5.2.1.1 Command Form**

498 `show <CIM_MemberOfCollection multiple instances>`

499 6.5.2.1.2 CIM Requirements

500 See CIM_MemberOfCollection in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
501 properties.

502 6.5.2.1.3 Behavior Requirements

503 6.5.2.1.3.1 Preconditions

504 \$instance represents the instance of CIM_RedundancySet which is referenced by
505 CIM_MemberOfCollection.

506 6.5.2.1.3.2 Pseudo Code

```
507 &smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.GetObjectPath() );  
508 &smEnd;
```

509 6.5.2.2 Show Command Form for a Single Instance – CIM_Fan Reference

510 This command form is used to show a single instance of CIM_MemberOfCollection. This command form
511 corresponds to a `show` command issued against a single instance of CIM_MemberOfCollection where
512 only one reference is specified and the reference is to the instance of CIM_Fan.

513 6.5.2.2.1 Command Form

```
514 show <CIM_MemberOfCollection single instance>
```

515 6.5.2.2.2 CIM Requirements

516 See CIM_MemberOfCollection in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
517 properties.

518 6.5.2.2.3 Behavior Requirements

519 6.5.2.2.3.1 Preconditions

520 \$instance represents the instance of CIM_Fan which is referenced by CIM_MemberOfCollection.

521 6.5.2.2.3.2 Pseudo Code

```
522 &smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.GetObjectPath() );  
523 &smEnd;
```

524 6.5.2.3 Show Command Form for a Single Instance Target – Both References

525 This command form is for the `show` verb applied to a single instance. This command form corresponds to
526 a `show` command issued against CIM_MemberOfCollection where both references are specified and
527 therefore the desired instance is unambiguously identified.

528 6.5.2.3.1 Command Form

```
529 show <CIM_MemberOfCollection single instance>
```

530 6.5.2.3.2 CIM Requirements

531 See CIM_MemberOfCollection in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
532 properties.

533 **6.5.2.3.3 Behavior Requirements**

534 **6.5.2.3.3.1 Preconditions**

535 \$instanceA represents the referenced instance of CIM_Fan through CIM_MemberOfCollection
536 association.

537 \$instanceB represents the instance of CIM_RedundancySet which is referenced by
538 CIM_MemberOfCollection.

539 **6.5.2.3.3.2 Pseudo Code**

```
540 &smShowAssociationInstance ( "CIM_MemberOfCollection", $instanceA.getObjectPath(),
541     $instanceB.getObjectPath() );
542 &smEnd;
```

543 **6.6 CIM_Fan**

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

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

550 **Table 6 – Command Verb Requirements for CIM_Fan**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.6.2.
Set	May	See 6.6.3.
Show	Shall	See 6.6.4.
Start	May	See 6.6.5.
Stop	May	See 6.6.6.

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

552 **6.6.1 Ordering of Results**

553 When results are returned for multiple instances of CIM_Fan, implementations shall utilize the following
554 algorithm to produce the natural (that is, default) ordering:

- 555 • Results for CIM_Fan are unordered; therefore, no algorithm is defined.

556 **6.6.2 Reset**

557 This section describes how to implement the `reset` verb when applied to an instance of CIM_Fan.
558 Implementations may support the use of the `reset` verb with CIM_Fan.

559 6.6.2.1 Command Form

```
560 reset <CIM_Fan single instance>
```

561 6.6.2.2 CIM Requirements

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

568 6.6.2.3 Behavior Requirements

569 6.6.2.3.1 Preconditions

570 \$instance represents the targeted instance of CIM_Fan.

```
571 $instance=<CIM_Fan single instance>;
```

572 6.6.2.3.1.1 Pseudo Code

```
573 &smResetRSC ( $instance.getObjectPath() );
574 &smEnd;
```

575 6.6.3 Set

576 This section describes how to implement the `set` verb when it is applied to an instance of `CIM_Fan`.
577 Implementations may support the use of the `set` verb with `CIM_Fan`.

578 The `set` verb is used to modify descriptive properties of the `CIM_Fan` instance.

579 6.6.3.1 General Usage of Set for a Single Property

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

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

585 6.6.3.1.1 Command Form

```
586 set <CIM_Fan single instance> <propertyname>=<propertyvalue>
```

587 6.6.3.1.2 CIM Requirements

588 See `CIM_Fan` in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory properties.

589 6.6.3.1.3 Behavior Requirements

```
590 $instance=<CIM_Fan single instance>
591 #propertyName[] = {<propertyname>};
592 #propertyValues[] = {<propertyvalue>};
593 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
594 &smEnd;
```

6.6.3.2 General Usage of Set for Multiple Properties

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

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

6.6.3.2.1 Command Form

```
set <CIM_Fan single instance> <propertyname1>=<propertyvalue1>
  <propertynamen>=<propertyvaluen>
```

6.6.3.2.2 CIM Requirements

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

6.6.3.2.3 Behavior Requirements

```
$instance=<CIM_Fan single instance>
#propertyName[] = {<propertyname>};
for #i < n
{
  #propertyName[#i] = <propertyname#i>
  #propertyValues[#i] = <propertyvalue#i>
}
&smSetInstance ( $instance, #propertyName[], #propertyValues[] );
&smEnd;
```

6.6.3.3 Setting the DesiredSpeed Property

This section describes how to set the speed of a fan.

6.6.3.3.1 Command Form

```
set <CIM_Fan single instance> desiredspeed=<requested value>
```

6.6.3.3.2 CIM Requirements

```
uint64 DesiredSpeed;
uint32 CIM_Fan.SetSpeed (
  [IN] uint64 DesiredSpeed );
```

6.6.3.3.3 Behavior Requirements

6.6.3.3.3.1 Preconditions

`$instance` represents the targeted instance of `CIM_Fan`.

```
$instance=<CIM_Fan single instance>;
#desiredSpeed=<requested value>
```

630 6.6.3.3.3.2 Pseudo Code

```
631 %InArguments[] = {newArgument("DesiredSpeed", #desiredSpeed) };
632         %OutArguments[] = {};
633 #Error = InvokeMethod ($instance.getObjectPath(),
634         "SetSpeed",
635         %InArguments[],
636         %OutArguments[],
637         #returnStatus);
638 if (0 != #Error.code) {
639 //method invocation failed
640     if ( (null != #Error.$error) && (null != #Error.$error[0]) )    {
641         //if the method invocation contains an embedded error
642         //use it for the Error for the overall job
643         &smAddError($job, #Error.$error[0]);
644         &smMakeCommandStatus($job);
645         &smEnd;
646     }
647     else if ( 17 == #Error.code ) {
648         //17 - CIM_ERR_METHOD_NOT_FOUND
649         // The specified extrinsic method does not exist.
650         $OperationError = smNewInstance("CIM_Error");
651         // CIM_ERR_METHOD_NOT_FOUND
652         $OperationError.CIMStatusCode = 17;
653         //Software Error
654         $OperationError.ErrorType = 10;
655         //Unknown
656         $OperationError.PerceivedSeverity = 0;
657         $OperationError.OwningEntity = DMTF:SMCLP;
658         $OperationError.MessageID = 0x00000001;
659         $OperationError.Message = "Operation is not supported."
660         &smAddError($job, $OperationError);
661         &smMakeCommandStatus($job);
662         &smEnd;
663     }
664     else {
665         //operation failed, but no detailed error instance, need to make one up
666         //make an Error instance and associate with job for Operation
667         $OperationError = smNewInstance("CIM_Error");
668         //CIM_ERR_FAILED
669         $OperationError.CIMStatusCode = 1;
670         //Software Error
671         $OperationError.ErrorType = 4;
672         //Unknown
673         $OperationError.PerceivedSeverity = 0;
674         $OperationError.OwningEntity = DMTF:SMCLP;
675         $OperationError.MessageID = 0x00000009;
676         $OperationError.Message = "An internal software error has occurred.";
677         &smAddError($job, $OperationError);
678         &smMakeCommandStatus($job);
```



```
679     &smEnd;
680   }
681 }//if CIM op failed
682 else if (0 == #returnStatus) {
683     //completed successfully
684     &smCommandCompleted($job);
685     &smEnd;
686 }
687 else if (1 == #returnStatus) {
688     //unsupported
689     $OperationError = smNewInstance("CIM_Error");
690     //CIM_ERR_NOT_SUPPORTED
691     $OperationError.CIMStatusCode = 7;
692     //Other
693     $OperationError.ErrorType = 1;
694     //Low
695     $OperationError.PerceivedSeverity = 2;
696     $OperationError.OwningEntity = DMTF:SMCLP;
697     $OperationError.MessageID = 0x00000001;
698     $OperationError.Message = "Operation is not supported.";
699     &smAddError($job, $OperationError);
700     &smMakeCommandStatus($job);
701     &smEnd;
702 }
703 else if (2 == #returnStatus) {
704     //generic failure
705     $OperationError = smNewInstance("CIM_Error");
706     //CIM_ERR_FAILED
707     $OperationError.CIMStatusCode = 1;
708     //Other
709     $OperationError.ErrorType = 1;
710     //Low
711     $OperationError.PerceivedSeverity = 2;
712     $OperationError.OwningEntity = DMTF:SMCLP;
713     $OperationError.MessageID = 0x00000002;
714     $OperationError.Message = "Failed. No further information is available.";
715     &smAddError($job, $OperationError);
716     &smMakeCommandStatus($job);
717 }
718 else {
719     // generic failure
720     $OperationError = smNewInstance("CIM_Error");
721     //CIM_ERR_FAILED
722     $OperationError.CIMStatusCode = 1;
723     //Other
724     $OperationError.ErrorType = 1;
725     //Low
726     $OperationError.PerceivedSeverity = 2;
727     $OperationError.OwningEntity = DMTF:SMCLP;
```

```

728     $OperationError.MessageID = 0x00000002;
729     $OperationError.Message = "Failed. No further information is available.";
730     &smAddError($job, $OperationError);
731     &smMakeCommandStatus($job);
732     &smEnd;
733 }
734 &smDisplayInstance ( $instance );
735 &smEnd;

```

736 6.6.4 Show

737 This section describes how to implement the `show` verb when applied to an instance of `CIM_Fan`.
 738 Implementations shall support the use of the `show` verb with `CIM_Fan`.

739 6.6.4.1 Show Command Form for Multiple Instances Target

740 This command form is used to show many instances of `CIM_Fan`.

741 6.6.4.1.1 Command Form

```
742 show <CIM_Fan multiple instances>
```

743 6.6.4.1.2 CIM Requirements

744 See `CIM_Fan` in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory properties.

745 6.6.4.1.3 Behavior Requirements

746 6.6.4.1.3.1 Preconditions

747 `$containerInstance` represents the instance of `CIM_ComputerSystem` which represents the
 748 container system and is associated to the targeted instances of `CIM_Fan` through the `CIM_SystemDevice`
 749 association.

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

751 6.6.4.1.3.2 Pseudo Code

```

752 #propertylist[] = NULL;
753 if ( false == #all)
754 {
755     #propertylist[] = <array of mandatory non-key property names (see CIM
756     Requirements)>;
757 }
758 &smShowInstances ( "CIM_Fan", "CIM_SystemDevice", $containerInstance.getObjectPath(),
759     #propertylist[] );
760 &smEnd;

```

761 6.6.4.2 Show Command Form for a Single Instance Target

762 This command form is used to show a single instance of `CIM_Fan`.

763 6.6.4.2.1 Command Form

```
764 show <CIM_Fan single instance>
```

765 6.6.4.2.2 CIM Requirements

766 See CIM_Fan in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory properties.

767 6.6.4.2.3 Behavior Requirements

768 6.6.4.2.3.1 Preconditions

769 \$instance represents the targeted instance of CIM_Fan.

```
770 $instance=<CIM_Fan single instance>;
```

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

772 6.6.4.2.3.2 Pseudo Code

```
773 #propertylist[] = NULL;
774 if ( false == #all)
775     {
776         #propertylist[] = <array of mandatory non-key property names (see CIM
777             Requirements)>;
778     }
779 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
780 &smEnd;
```

781 6.6.5 Start

782 This section describes how to implement the `start` verb when applied to an instance of CIM_Fan.

783 Implementations may support the use of the `start` verb with CIM_Fan.

784 6.6.5.1.1 Command Form

```
785 start <CIM_Fan single instance>
```

786 6.6.5.1.2 CIM Requirements

```
787 uint16 EnabledState;
788 uint16 RequestedState;
789 uint32 CIM_Fan.RequestStateChange (
790     [IN] uint16 RequestedState,
791     [OUT] REF CIM_ConcreteJob Job,
792     [IN] datetime TimeoutPeriod );
```

793 6.6.5.1.3 Behavior Requirements

794 6.6.5.1.3.1 Preconditions

795 \$instance represents the targeted instance of CIM_Fan.

```
796 $instance=<CIM_Fan single instance>;
```

797 6.6.5.1.3.2 Pseudo Code

```
798 &smStartRSC ( $instance.getObjectPath() );
799 &smEnd;
```

800 **6.6.6 Stop**

801 This section describes how to implement the `stop` verb when applied to an instance of `CIM_Fan`.
 802 Implementations may support the use of the `stop` verb with `CIM_Fan`.

803 **6.6.6.1.1 Command Form**

804 `stop <CIM_Fan single instance>`

805 **6.6.6.1.2 CIM Requirements**

```
806 uint16 EnabledState;
807 uint16 RequestedState;
808 uint32 CIM_Fan.RequestStateChange (
809     [IN] uint16 RequestedState,
810     [OUT] REF CIM_ConcreteJob Job,
811     [IN] datetime TimeoutPeriod );
```

812 **6.6.6.1.3 Behavior Requirements**813 **6.6.6.1.3.1 Preconditions**

814 `$instance` represents the targeted instance of `CIM_Fan`.

815 `$instance=<CIM_Fan single instance>;`

816 **6.6.6.1.3.2 Pseudo Code**

```
817 &smStopRSC ( $instance.getObjectPath() );
818 &smEnd;
```

819 **6.7 CIM_RedundancySet**

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

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

826 **Table 7 – Command Verb Requirements for CIM_RedundancySet**

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

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

828 6.7.1 Ordering of Results

829 When results are returned for multiple instances of CIM_RedundancySet, implementations shall utilize
830 the following algorithm to produce the natural (that is, default) ordering:

- 831 • Results for CIM_RedundancySet are unordered; therefore, no algorithm is defined.

832 6.7.2 Set

833 This section describes how to implement the `set` verb when it is applied to an instance of
834 CIM_RedundancySet. Implementations may support the use of the `set` verb with CIM_RedundancySet.

835 The `set` verb is used to modify descriptive properties of the CIM_RedundancySet instance.

836 6.7.2.1 Set Command for Failovers

837 This section describes on how to use the `set` verb when it is applied to an instance of
838 CIM_RedundancySet to failover from an active fan to a spare fan.

839 6.7.2.1.1 Command Form

```
840 set <CIM_RedundancySet single instance> failoverfrom=<CIM_Fan single instance>
841     failovertto=<CIM_Fan single instance>
```

842 6.7.2.1.2 CIM Requirements

```
843 uint32 CIM_RedundancySet.Failover (
844     [IN] REF CIM_ManagedElement FailoverFrom,
845     [IN] REF CIM_ManagedElement FailoverTo);
```

846 6.7.2.1.3 Behavior Requirements

847 6.7.2.1.3.1 Preconditions

```
848 $instance=<CIM_RedundancySet single instance>
849 $FailoverFrom=<failoverfrom requested instance of CIM_Fan>
850 $FailoverTo=<failovertto requested instance of CIM_Fan>
```

851 6.7.2.1.3.2 Pseudo Code

```
852 %InArguments[] = { newArgument ( "FailoverFrom", $FailoverFrom.getObjectPath() ),
853     newArgument ( "FailoverTo", $FailoverTo.getObjectPath() ) };
854 %OutArguments[] = {};
855 #Error = InvokeMethod ($target->,
856     "Failover",
857     %InArguments[],
858     %OutArguments[],
859     #returnStatus);
860 if (0 != #Error.code)
861 {
862     //method invocation failed
863     if ( (null != #Error.$error) && (null != #Error.$error[0]) )
864     {
865         //if the method invocation contains an embedded error
866         //use it for the Error for the overall job
867         &smAddError($job, #Error.$error[0]);
868         &smMakeCommandStatus($job);
```

```
869     &smEnd;
870 }
871 else if ( 17 == #Error.code ) {
872     //17 - CIM_ERR_METHOD_NOT_FOUND
873     // The specified extrinsic method does not exist.
874     $OperationError = smNewInstance("CIM_Error");
875     // CIM_ERR_METHOD_NOT_FOUND
876     $OperationError.CIMStatusCode = 17;
877     //Software Error
878     $OperationError.ErrorType = 10;
879     //Unknown
880     $OperationError.PerceivedSeverity = 0;
881     $OperationError.OwningEntity = DMTF:SMCLP;
882     $OperationError.MessageID = 0x00000001;
883     $OperationError.Message = "Operation is not supported."
884     &smAddError($job, $OperationError);
885     &smMakeCommandStatus($job);
886     &smEnd;
887 }
888 else
889 {
890     //operation failed, but no detailed error instance, need to make one up
891     //make an Error instance and associate with job for Operation
892     $OperationError = smNewInstance("CIM_Error");
893     //CIM_ERR_FAILED
894     $OperationError.CIMStatusCode = 1;
895     //Software Error
896     $OperationError.ErrorType = 4;
897     //Unknown
898     $OperationError.PerceivedSeverity = 0;
899     $OperationError.OwningEntity = DMTF:SMCLP;
900     $OperationError.MessageID = 0x00000009;
901     $OperationError.Message = "An internal software error has occurred.";
902     &smAddError($job, $OperationError);
903     &smMakeCommandStatus($job);
904     &smEnd;
905 }
906 }//if CIM op failed
907 else if (0 == #returnStatus) {
908     //completed successfully
909     &smCommandCompleted($job);
910     &smEnd;
911 }
912 else if (1 == #returnStatus) {
913     //unsupported
914     $OperationError = smNewInstance("CIM_Error");
915     //CIM_ERR_NOT_SUPPORTED
916     $OperationError.CIMStatusCode = 7;
917     //Other
```

```

918     $OperationError.ErrorType = 1;
919     //Low
920     $OperationError.PerceivedSeverity = 2;
921     $OperationError.OwningEntity = DMTF:SMCLP;
922     $OperationError.MessageID = 0x00000001;
923     $OperationError.Message = "Operation is not supported.";
924     &smAddError($job, $OperationError);
925     &smMakeCommandStatus($job);
926     &smEnd;
927 }
928 else if (2 == #returnStatus) {
929     //generic failure
930     $OperationError = smNewInstance("CIM_Error");
931     //CIM_ERR_FAILED
932     $OperationError.CIMStatusCode = 1;
933     //Other
934     $OperationError.ErrorType = 1;
935     //Low
936     $OperationError.PerceivedSeverity = 2;
937     $OperationError.OwningEntity = DMTF:SMCLP;
938     $OperationError.MessageID = 0x00000002;
939     $OperationError.Message = "Failed. No further information is available.";
940     &smAddError($job, $OperationError);
941     &smMakeCommandStatus($job);
942 }
943 else {
944     //unspecified return code, generic failure
945     $OperationError = smNewInstance("CIM_Error");
946     //CIM_ERR_FAILED
947     $OperationError.CIMStatusCode = 1;
948     //Other
949     $OperationError.ErrorType = 1;
950     //Low
951     $OperationError.PerceivedSeverity = 2;
952     $OperationError.OwningEntity = DMTF:SMCLP;
953     $OperationError.MessageID = 0x00000002;
954     $OperationError.Message = "Failed. No further information is available.";
955     &smAddError($job, $OperationError);
956     &smMakeCommandStatus($job);
957     &smEnd;
958 }

```

959 6.7.2.2 General Usage of Set for a Single Property

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

962 The requirement for supporting modification of a property using this command form shall be equivalent to
963 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
964 in the [Fan Profile](#).

965 **6.7.2.2.1 Command Form**

```
966 set <CIM_RedundancySet single instance> <propertyname>=<propertyvalue>
```

967 **6.7.2.2.2 CIM Requirements**

968 See CIM_RedundancySet in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
969 properties.

970 **6.7.2.2.3 Behavior Requirements**

```
971 $instance=<CIM_RedundancySet single instance>
972 #propertyNames[] = {<propertyname>;};
973 #propertyValues[] = {<propertyvalue>;};
974 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
975 &smEnd;
```

976 **6.7.2.3 General Usage of Set for Multiple Properties**

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

980 The requirement for supporting modification of a property using this command form shall be equivalent to
981 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
982 in the [Fan Profile](#).

983 **6.7.2.3.1 Command Form**

```
984 set <CIM_RedundancySet single instance> <propertyname1>=<propertyvalue1>
985 <propertyname2>=<propertyvalue2>
```

986 **6.7.2.3.2 CIM Requirements**

987 See CIM_RedundancySet in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
988 properties.

989 **6.7.2.3.3 Behavior Requirements**

```
990 $instance=<CIM_RedundancySet single instance>
991 #propertyNames[] = {<propertyname>;};
992 for #i < n
993 {
994     #propertyNames[#i] = <propertyname#i>
995     #propertyValues[#i] = <propertyvalue#i>
996 }
997 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
998 &smEnd;
```

999 **6.7.3 Show**

1000 This section describes how to implement the `show` verb when applied to an instance of
1001 CIM_RedundancySet. Implementations shall support the use of the `show` verb with CIM_RedundancySet.

1002 **6.7.3.1 Show Command Form for Multiple Instances Target**

1003 This command form is used to show many instances of CIM_RedundancySet.

1004 **6.7.3.1.1 Command Form**1005 `show <CIM_RedundancySet multiple instances>`1006 **6.7.3.1.2 CIM Requirements**1007 See CIM_RedundancySet in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
1008 properties.1009 **6.7.3.1.3 Behavior Requirements**1010 **6.7.3.1.3.1 Preconditions**1011 \$containerInstance represents the instance of CIM_ComputerSystem which represents the
1012 container system and is associated to the targeted instances of CIM_RedundancySet through the
1013 CIM_OwningCollectionElement association.

1014 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1015 **6.7.3.1.3.2 Pseudo Code**

```

1016 #propertylist[] = NULL;
1017 if ( false == #all)
1018     {
1019         #propertylist[] = <array of mandatory non-key property names (see CIM
1020             Requirements)>;
1021     }
1022 &smShowInstances ( "CIM_RedundancySet", "CIM_OwningCollectionElement",
1023     $containerInstance.getObjectPath(), #propertylist[] );
1024 &smEnd;

```

1025 **6.7.3.2 Show Command Form for a Single Instance Target**

1026 This command form is used to show a single instance of CIM_RedundancySet.

1027 **6.7.3.2.1 Command Form**1028 `show <CIM_RedundancySet single instance>`1029 **6.7.3.2.2 CIM Requirements**1030 See CIM_RedundancySet in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
1031 properties.1032 **6.7.3.2.3 Behavior Requirements**1033 **6.7.3.2.3.1 Preconditions**

1034 \$instance represents the targeted instance of CIM_RedundancySet.

1035 `$instance=<CIM_RedundancySet single instance>;`

1036 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1037 **6.7.3.2.3.2 Pseudo Code**

```

1038 #propertylist[] = NULL;
1039 if ( false == #all)
1040     {
1041         #propertylist[] = <array of mandatory non-key property names (see CIM
1042             Requirements)>;
1043     }
1044 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1045 &smEnd;

```

1046 **6.8 CIM_SystemDevice**

1047 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1048 Table 8 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1049 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1050 target. Table 8 is for informational purposes only; in case of a conflict between Table 8 and requirements
 1051 detailed in the following sections, the text detailed in the following sections supersedes the information in
 1052 Table 8.

1053 **Table 8 – Command Verb Requirements for CIM_SystemDevice**

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

1054 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
 1055 `reset`, `set`, `start`, and `stop`.

1056 **6.8.1 Ordering of Results**

1057 When results are returned for multiple instances of `CIM_SystemDevice`, implementations shall utilize the
 1058 following algorithm to produce the natural (that is, default) ordering:

- 1059 • Results for `CIM_SystemDevice` are unordered; therefore, no algorithm is defined.

1060 **6.8.2 Show**

1061 This section describes how to implement the `show` verb when applied to an instance of
 1062 `CIM_SystemDevice`. Implementations shall support the use of the `show` verb with `CIM_SystemDevice`.

1063 **6.8.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference**

1064 This command form is used to show many instances of CIM_SystemDevice. This command form
1065 corresponds to a `show` command issued against the instance of CIM_SystemDevice where only one
1066 reference is specified and the reference is to the scoping instance of CIM_ComputerSystem.

1067 **6.8.2.1.1 Command Form**

```
1068 show <CIM_SystemDevice multiple instances>
```

1069 **6.8.2.1.2 CIM Requirements**

1070 See CIM_SystemDevice in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
1071 properties.

1072 **6.8.2.1.3 Behavior Requirements**

1073 **6.8.2.1.3.1 Preconditions**

1074 `$instance` represents the instance of a CIM_ComputerSystem, which is referenced by
1075 CIM_SystemDevice.

1076 **6.8.2.1.3.2 Pseudo Code**

```
1077 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath() );  
1078 &smEnd;
```

1079 **6.8.2.2 Show Command Form for a Single Instance Target – CIM_Fan Reference**

1080 This command form is used to show a single instance of CIM_SystemDevice. This command form
1081 corresponds to a `show` command issued against a single instance of CIM_SystemDevice, where only one
1082 reference is specified and the reference is to the instance of CIM_Fan.

1083 **6.8.2.2.1 Command Form**

```
1084 show <CIM_SystemDevice single instance>
```

1085 **6.8.2.2.2 CIM Requirements**

1086 See CIM_RedundancySet in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
1087 properties.

1088 **6.8.2.2.3 Behavior Requirements**

1089 **6.8.2.2.3.1 Preconditions**

1090 `$instance` represents the instance of CIM_Fan which is referenced by the CIM_SystemDevice
1091 association.

1092 **6.8.2.2.3.2 Pseudo Code**

```
1093 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath() );  
1094 &smEnd;
```

1095 **6.8.2.3 Show Command Form for a Single Instance Target – Both References**

1096 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1097 a `show` command issued against CIM_SystemDevice where both references are specified and therefore
1098 the desired instance is unambiguously identified.

1099 **6.8.2.3.1 Command Form**1100 `show <CIM_SystemDevice single instance>`1101 **6.8.2.3.2 CIM Requirements**1102 See CIM_RedundancySet in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
1103 properties.1104 **6.8.2.3.3 Behavior Requirements**1105 **6.8.2.3.3.1 Preconditions**1106 `$instanceA` represents the referenced instance of CIM_Fan through CIM_SystemDevice association.1107 `$instanceB` represents the instance of CIM_ComputerSystem which is referenced by
1108 CIM_SystemDevice.1109 **6.8.2.3.3.2 Pseudo Code**1110 `&smShowAssociationInstance ("CIM_SystemDevice", $instanceA.getObjectPath(),`
1111 `$instanceB.getObjectPath());`
1112 `&smEnd;`1113 **6.9 CIM_OwningCollectionElement**1114 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).1115 Table 9 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1116 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1117 target. Table 9 is for informational purposes only; in case of a conflict between Table 9 and requirements
1118 detailed in the following sections, the text detailed in the following sections supersedes the information in
1119 Table 9.1120 **Table 9 – Command Verb Requirements for CIM_OwningCollectionElement**

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	

1121 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1122 `reset`, `set`, `start`, and `stop`.

1123 6.9.1 Ordering of Results

1124 When results are returned for multiple instances of CIM_OwningCollectionElement, implementations shall
1125 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1126 • Results for CIM_OwningCollectionElement are unordered; therefore, no algorithm is defined.

1127 6.9.2 Show

1128 This section describes how to implement the `show` verb when applied to an instance of
1129 CIM_OwningCollectionElement. Implementations shall support the use of the `show` verb with
1130 CIM_OwningCollectionElement.

1131 6.9.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

1132 This command form is used to show many instances of CIM_OwningCollectionElement. This command
1133 form corresponds to a `show` command issued against the instance of CIM_OwningCollectionElement
1134 where only one reference is specified and the reference is to the scoping instance of
1135 CIM_ComputerSystem.

1136 6.9.2.1.1 Command Form

```
1137 show <CIM_OwningCollectionElement multiple instances>
```

1138 6.9.2.1.2 CIM Requirements

1139 See CIM_OwningCollectionElement in the “CIM Elements” section of the [Fan Profile](#) for the list of
1140 mandatory properties.

1141 6.9.2.1.3 Behavior Requirements

1142 6.9.2.1.3.1 Preconditions

1143 `$instance` represents the instance of a CIM_ComputerSystem, which is referenced by
1144 CIM_OwningCollectionElement.

1145 6.9.2.1.3.2 Pseudo Code

```
1146 &smShowAssociationInstances ( "CIM_OwningCollectionElement",  
1147     $instance.getObjectPath() );  
1148 &smEnd;
```

1149 6.9.2.2 Show Command Form for a Single Instance Target – CIM_RedundancySet Reference

1150 This command form is used to show a single instance of CIM_OwningCollectionElement. This command
1151 form corresponds to a `show` command issued against a single instance of
1152 CIM_OwningCollectionElement, where only one reference is specified and the reference is to an instance
1153 of CIM_RedundancySet.

1154 6.9.2.2.1 Command Form

```
1155 show <CIM_OwningCollectionElement single instance>
```

1156 6.9.2.2.2 CIM Requirements

1157 See CIM_RedundancySet in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
1158 properties.

1159 **6.9.2.2.3 Behavior Requirements**1160 **6.9.2.2.3.1 Preconditions**

1161 \$instance represents the instance of CIM_RedundancySet which is referenced by
1162 CIM_OwningCollectionElement.

1163 **6.9.2.2.3.2 Pseudo Code**

```
1164 &smShowAssociationInstances ( "CIM_OwningCollectionElement",  
1165     $instance.getObjectPath() );  
1166 &smEnd;
```

1167 **6.9.2.3 Show Command Form for a Single Instance Target – Both References**

1168 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1169 a `show` command issued against `CIM_OwningCollectionElement` where both references are specified and
1170 therefore the desired instance is unambiguously identified.

1171 **6.9.2.3.1 Command Form**

```
1172 show <CIM_OwningCollectionElement single instance>
```

1173 **6.9.2.3.2 CIM Requirements**

1174 See `CIM_RedundancySet` in the “CIM Elements” section of the [Fan Profile](#) for the list of mandatory
1175 properties.

1176 **6.9.2.3.3 Behavior Requirements**1177 **6.9.2.3.3.1 Preconditions**

1178 \$instanceA represents the referenced instance of `CIM_RedundancySet` through
1179 `CIM_OwningCollectionElement` association.

1180 \$instanceB represents the instance of `CIM_ComputerSystem` which is referenced by
1181 `CIM_OwningCollectionElement`.

1182 **6.9.2.3.3.2 Pseudo Code**

```
1183 &smShowAssociationInstance ( "CIM_OwningCollectionElement",  
1184     $instanceA.getObjectPath(), $instanceB.getObjectPath() );  
1185 &smEnd;
```

1186

ANNEX A
(informative)

Change Log

1187
1188
1189
1190
1191

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

1192