



1
2
3
4

Document Number: DSP0826

Date: 2009-07-14

Version: 1.0.0

5 **Software Inventory 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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

CONTENTS

31	Foreword	5
32	Introduction	6
33	1 Scope	7
34	2 Normative References.....	7
35	2.1 Approved References	7
36	2.2 Other References.....	7
37	3 Terms and Definitions.....	7
38	4 Symbols and Abbreviated Terms.....	8
39	5 Recipes.....	9
40	6 Mappings.....	9
41	6.1 CIM_SoftwareIdentity.....	9
42	6.2 CIM_InstalledSoftwareIdentity	13
43	6.3 CIM_ElementSoftwareIdentity	16
44	6.4 CIM_SystemSpecificCollection	19
45	6.5 CIM_HostedCollection	22
46	6.6 CIM_MemberOfCollection	25
47	6.7 CIM_SoftwareIdentityResource	27
48	6.8 CIM_SAPAvailableForElement.....	29
49	6.9 CIM_HostedAccessPoint.....	32
50	6.10 CIM_OrderedComponent	35
51	6.11 CIM_OrderedDependency.....	37
52	ANNEX A (informative) Change Log	40
53		

54 Tables

55	Table 1 – Command Verb Requirements for CIM_SoftwareIdentity.....	9
56	Table 2 – Command Verb Requirements for CIM_InstalledSoftwareIdentity	13
57	Table 3 – Command Verb Requirements for CIM_ElementSoftwareIdentity	16
58	Table 4 – Command Verb Requirements for CIM_SystemSpecificCollection.....	19
59	Table 5 – Command Verb Requirements for CIM_HostedCollection	22
60	Table 6 – Command Verb Requirements for CIM_MemberOfCollection	25
61	Table 7 – Command Verb Requirements for CIM_SoftwareIdentityResource	28
62	Table 8 – Command Verb Requirements for CIM_SAPAvailableForElement.....	30
63	Table 9 – Command Verb Requirements for CIM_HostedAccessPoint	32
64	Table 10 – Command Verb Requirements for CIM_OrderedComponent	35
65	Table 11 – Command Verb Requirements for CIM_OrderedDependency.....	38
66		

68

Foreword

69 The *Software Inventory Profile SM CLP Mapping Specification* (DSP0826) was prepared by the Server
70 Management Working Group.

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

73 **Conventions**

74 The pseudo-code conventions utilized in this document are the Recipe Conventions as defined in SNIA
75 [SMI-S V1.1.0](#), section 7.6.

76 **Acknowledgements**

77 The authors wish to acknowledge the following participants from the DMTF Server Management Working
78 Group:

- 79 • RadhaKrishna R. Dasari – Dell
- 80 • Khachatur Papanyan – Dell
- 81 • Perry Vincent – Intel
- 82 • Aaron Merkin – IBM

83

84

Introduction

85 This document defines the SM CLP mapping for CIM elements described in the [Software Inventory](#)
86 [Profile](#). The information in this specification, combined with *SM CLP-to-CIM Common Mapping*
87 *Specification V1.0* ([DSP0216](#)), is intended to be sufficient to implement SM CLP commands relevant to
88 the classes, properties, and methods described in the [Software Inventory Profile](#) using CIM operations.

89 The target audience for this specification is implementers of the SM CLP support for the [Software](#)
90 [Inventory Profile](#).

91

92 Software Inventory Profile SM CLP Mapping Specification

93 1 Scope

94 This specification contains the requirements for an implementation of the SM CLP to provide access to,
95 and implement the behaviors of, the [Software Inventory 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 DSP1023, *Software Inventory Profile 1.0*,
104 http://www.dmtf.org/standards/published_documents/DSP1023_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 *SM CLP-to-CIM Common Mapping Specification 1.0* ([DSP0216](#)).

- 179 • smShowInstance
 180 • smShowInstances
 181 • smShowAssociationInstance
 182 • smShowAssociationInstances

183 There are no Local Recipes defined for use in this mapping.

184 **6 Mappings**

185 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
 186 the *Software Inventory Profile*. Requirements specified here related to support for a CLP verb for a
 187 particular class are solely within the context of this profile.

188 **6.1 CIM_SoftwareIdentity**

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

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

195 **Table 1 – Command Verb Requirements for CIM_SoftwareIdentity**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	

Command Verb	Requirement	Comments
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.1.2.
start	Not supported	
stop	Not supported	

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

198 6.1.1 Ordering of Results

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

- 201 • Results for CIM_SoftwareIdentity are unordered; therefore, no algorithm is defined.

202 6.1.2 Show

203 This section describes how to implement the `show` verb when applied to an instance of
204 CIM_SoftwareIdentity. Implementations shall support the use of the `show` verb with
205 CIM_SoftwareIdentity.

206 6.1.2.1 Show Command Form for a Single Object Target – CIM_SoftwareIdentity

207 This command form is used to show a single instance of CIM_SoftwareIdentity. This command form
208 corresponds to a `show` command issued against a single instance of CIM_SoftwareIdentity where only
209 one reference is specified and the reference is to an instance of CIM_SoftwareIdentity.

210 6.1.2.1.1 Command Form

```
211 show <CIM_SoftwareIdentity single object>
```

212 6.1.2.1.2 CIM Requirements

213 See CIM_SoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
214 mandatory properties.

215 6.1.2.1.3 Behavior Requirements

216 6.1.2.1.3.1 Preconditions

217 `$instance` represents the targeted instance of CIM_SoftwareIdentity.

```
218 $instance=<CIM_SoftwareIdentity single object>;
```

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

220 6.1.2.1.3.2 Pseudo Code

```

221 #propertylist[] = NULL;
222 if ( false == #all )
223     {
224         #propertylist[] = {<array of mandatory non-key property names (see CIM
225             Requirements)>}
226     }
227 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
228 &smEnd;

```

229 6.1.2.2 Show Command Form for Multiple Objects Target for the Installed Software

230 6.1.2.2.1 Command Form

```
231 show <CIM_SoftwareIdentity multiple objects>
```

232 6.1.2.2.2 CIM Requirements

233 See CIM_SoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
 234 mandatory properties.

235 6.1.2.2.3 Behavior Requirements

236 6.1.2.2.3.1 Preconditions

237 \$containerInstance is the instance of CIM_ComputerSystem or CIM_System that is associated with
 238 the targeted instances of CIM_SoftwareIdentity through the CIM_InstalledSoftwareIdentity association.

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

240 6.1.2.2.3.2 Pseudo Code

```

241 #propertylist[] = NULL;
242 if ( false == #all )
243     {
244         #propertylist[] = {<array of mandatory non-key property names (see CIM
245             Requirements)>}
246     }
247 &smShowInstances ( "CIM_SoftwareIdentity", "CIM_InstalledSoftwareIdentity",
248     $containerInstance.getObjectPath(), #propertylist[] );
249 &smEnd;

```

250 6.1.2.3 Show Command Form for Multiple Objects Target for the Available Software

251 6.1.2.3.1 Command Form

```
252 show <CIM_SoftwareIdentity multiple objects>
```

253 6.1.2.3.2 CIM Requirements

254 See CIM_SoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
 255 mandatory properties.

256 **6.1.2.3.3 Behavior Requirements**257 **6.1.2.3.3.1 Preconditions**

258 \$containerInstance contains the instance of CIM_SystemSpecificCollection that is associated with
259 the targeted instances of CIM_SoftwareIdentity through the CIM_MemberOfCollection association.

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

261 **6.1.2.3.3.2 Pseudo Code**

```
262 #propertylist[] = NULL;
263 if ( false == #all )
264     {
265         #propertylist[] = {<array of mandatory non-key property names (see CIM
266             Requirements)>}
267     }
268 &smShowInstances ( "CIM_SoftwareIdentity", "CIM_MemberOfCollection",
269     $containerInstance.getObjectPath(), #propertylist[] );
270 &smEnd;
```

271 **6.1.2.4 Show Command Form for Multiple Objects Target for a Software Bundle**272 **6.1.2.4.1 Command Form**

```
273 show <CIM_SoftwareIdentity multiple objects>
```

274 **6.1.2.4.2 CIM Requirements**

275 See CIM_SoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
276 mandatory properties.

277 **6.1.2.4.3 Behavior Requirements**278 **6.1.2.4.3.1 Preconditions**

279 \$containerInstance contains the instance of CIM_SoftwareIdentity that is associated with the
280 targeted instances of CIM_SoftwareIdentity through the CIM_OrderedComponent association.

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

282 **6.1.2.4.3.2 Pseudo Code**

```
283 #propertylist[] = NULL;
284 if ( false == #all )
285     {
286         #propertylist[] = {<array of mandatory non-key property names (see CIM
287             Requirements)>}
288     }
289 &smShowInstances ( "CIM_SoftwareIdentity", "CIM_OrderedComponent",
290     "GroupComponent", "PartComponent", $containerInstance.getObjectPath(), null,
291     #propertylist[] );
292 &smEnd;
```

293 **6.1.2.5 Show Command Form for Multiple Objects Target for a Software Dependency**

294 **6.1.2.5.1 Command Form**

295 `show <CIM_SoftwareIdentity multiple objects>`

296 **6.1.2.5.2 CIM Requirements**

297 See CIM_SoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
 298 mandatory properties.

299 **6.1.2.5.3 Behavior Requirements**

300 **6.1.2.5.3.1 Preconditions**

301 \$containerInstance contains the instance of CIM_SoftwareIdentity that is associated with the
 302 targeted instances of CIM_SoftwareIdentity through the CIM_OrderedDependency association.

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

304 **6.1.2.5.3.2 Pseudo Code**

```

305 #propertylist[] = NULL;
306 if ( false == #all )
307     {
308         #propertylist[] = {<array of mandatory non-key property names (see CIM
309             Requirements)>}
310     }
311 &smShowInstances ( "CIM_SoftwareIdentity", "CIM_OrderedDependency",
312     $containerInstance.getObjectPath(), #propertylist[] );
313 &smEnd;
```

314 **6.2 CIM_InstalledSoftwareIdentity**

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

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

321 **Table 2 – Command Verb Requirements for CIM_InstalledSoftwareIdentity**

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 6.2.2.
start	Not supported	
stop	Not supported	

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

324 6.2.1 Ordering of Results

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

- 327 • Results for CIM_InstalledSoftwareIdentity are unordered; therefore, no algorithm is defined.

328 6.2.2 Show

329 This section describes how to implement the `show` verb when applied to an instance of
 330 CIM_InstalledSoftwareIdentity. Implementations shall support the use of the `show` verb with
 331 CIM_InstalledSoftwareIdentity.

332 6.2.2.1 Show Command Form for Multiple Objects Target – CIM_System or 333 CIM_ComputerSystem Reference

334 This command form is used to show many instances of CIM_InstalledSoftwareIdentity. This command
 335 form corresponds to a `show` command issued against instances of CIM_InstalledSoftwareIdentity where
 336 only one reference is specified and the reference is to the scoping instance of CIM_System or
 337 CIM_ComputerSystem.

338 6.2.2.1.1 Command Form

```
339 show <CIM_InstalledSoftwareIdentity multiple objects>
```

340 6.2.2.1.2 CIM Requirements

341 See CIM_InstalledSoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the
 342 list of mandatory properties.

343 6.2.2.1.3 Behavior Requirements

344 6.2.2.1.3.1 Preconditions

345 `$instance` represents the instance of a CIM_System or CIM_ComputerSystem, which is referenced by
 346 CIM_InstalledSoftwareIdentity.

```
347 $instance=<CIM_ComputerSystem single object>;
```

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

349 6.2.2.1.3.2 Pseudo Code

```

350 #propertylist[] = NULL;
351 if ( false == #all )
352     {
353         #propertylist[] = <array of mandatory non-key property names (see CIM
354             Requirements)>;
355     }
356 &smShowAssociationInstances ( "CIM_InstalledSoftwareIdentity",
357     $instance.getObjectPath(), #propertylist[] );
358 &smEnd;

```

359 6.2.2.2 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

360 This command form is used to show multiple instances of CIM_InstalledSoftwareIdentity. This command
 361 form corresponds to a `show` command issued against instances of CIM_InstalledSoftwareIdentity where
 362 only one reference is specified and the reference is to the instance of CIM_SoftwareIdentity

363 6.2.2.2.1 Command Form

```

364 show <CIM_InstalledSoftwareIdentity multiple objects>

```

365 6.2.2.2.2 CIM Requirements

366 See CIM_InstalledSoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the
 367 list of mandatory properties.

368 6.2.2.2.3 Behavior Requirements

369 6.2.2.2.3.1 Preconditions

370 \$instance represents the instance of a CIM_SoftwareIdentity, which is referenced by
 371 CIM_InstalledSoftwareIdentity.

```

372 $instance=<CIM_SoftwareIdentity single object>

```

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

374 6.2.2.2.3.2 Pseudo Code

```

375 #propertylist[] = NULL;
376 if ( false == #all )
377     {
378         #propertylist[] = <array of mandatory non-key property names (see CIM
379             Requirements)>;
380     }
381 &smShowAssociationInstances ( "CIM_InstalledSoftwareIdentity",
382     $instance.getObjectPath(), #propertylist[] );
383 &smEnd;

```

384 6.2.2.3 Show Command Form for a Single Object Target – Both References

385 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 386 the `show` command issued against CIM_InstalledSoftwareIdentity where both references are specified
 387 and therefore the desired instance is unambiguously identified.

388 **6.2.2.3.1 Command Form**389 `show <CIM_InstalledSoftwareIdentity single object>`390 **6.2.2.3.2 CIM Requirements**391 See CIM_InstalledSoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the
392 list of mandatory properties.393 **6.2.2.3.3 Behavior Requirements**394 **6.2.2.3.3.1 Preconditions**395 \$instanceA represents the instance of CIM_System or CIM_ComputerSystem and \$instanceB
396 represents the instance of CIM_SoftwareIdentity, both of which are referenced by
397 CIM_InstalledSoftwareIdentity.398 `$instanceA=<CIM_ComputerSystem single object>;`
399 `$instanceB=<CIM_SoftwareIdentity single object>;`

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

401 **6.2.2.3.3.2 Pseudo Code**402 `#propertylist[] = NULL;`
403 `if (false == #all)`
404 `{`
405 `#propertylist[] = <array of mandatory non-key property names (see CIM`
406 `Requirements)>;`
407 `}`
408 `&smShowAssociationInstance ("CIM_InstalledSoftwareIdentity",`
409 `$instanceA.getObjectPath(), $instanceB.getObjectPath(), #propertylist[]);`
410 `&smEnd;`411 **6.3 CIM_ElementSoftwareIdentity**412 The cd, help, version, and exit verbs shall be supported as described in [DSP0216](#).413 Table 3 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
414 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
415 target. Table 3 is for informational purposes only; in case of a conflict between Table 3 and requirements
416 detailed in the following sections, the text detailed in the following sections supersedes the information in
417 Table 3.418 **Table 3 – Command Verb Requirements for CIM_ElementSoftwareIdentity**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	

Command Verb	Requirement	Comments
reset	Not supported	
set	Not supported	
show	Shall	See 6.3.2.
start	Not supported	
stop	Not supported	

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

421 6.3.1 Ordering of Results

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

- 424 • Results for CIM_ElementSoftwareIdentity are unordered; therefore, no algorithm is defined.

425 6.3.2 Show

426 This section describes how to implement the show verb when applied to an instance of
427 CIM_ElementSoftwareIdentity. Implementations shall support the use of the show verb with
428 CIM_ElementSoftwareIdentity.

429 6.3.2.1 Show Command Form for Multiple Objects Target – CIM_ManagedElement Reference

430 This command form is used to show many instances of CIM_ElementSoftwareIdentity. This command
431 form corresponds to a show command issued against instances of CIM_ElementSoftwareIdentity where
432 only one reference is specified and the reference is to the scoping instance of CIM_ManagedElement.

433 6.3.2.1.1 Command Form

```
434 show <CIM_ElementSoftwareIdentity multiple objects>
```

435 6.3.2.1.2 CIM Requirements

436 See CIM_ElementSoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the
437 list of mandatory properties.

438 6.3.2.1.3 Behavior Requirements

439 6.3.2.1.3.1 Preconditions

440 \$instance represents the instance of a CIM_ManagedElement, which is referenced by
441 CIM_ElementSoftwareIdentity.

```
442 $instance=<CIM_ManagedElement single object>;
```

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

444 **6.3.2.1.3.2 Pseudo Code**

```

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

```

454 **6.3.2.2 Show Command Form for Multiple Objects – CIM_SoftwareIdentity Reference**

455 This command form is used to show multiple instances of CIM_ElementSoftwareIdentity. This command
 456 form corresponds to a show command issued against instances of CIM_ElementSoftwareIdentity where
 457 only one reference is specified and the reference is to the instance of CIM_SoftwareIdentity.

458 **6.3.2.2.1 Command Form**

```

459 show <CIM_ElementSoftwareIdentity multiple objects>

```

460 **6.3.2.2.2 CIM Requirements**

461 See CIM_ElementSoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the
 462 list of mandatory properties.

463 **6.3.2.2.3 Behavior Requirements**464 **6.3.2.2.3.1 Preconditions**

465 \$instance represents the instance of a CIM_SoftwareIdentity, which is referenced by
 466 CIM_ElementSoftwareIdentity.

```

467 $instance=<CIM_SoftwareIdentity single object>

```

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

469 **6.3.2.2.3.2 Pseudo Code**

```

470 #propertylist[] = NULL;
471 if ( false == #all )
472     {
473         #propertylist[] = <array of mandatory non-key property names (see CIM
474             Requirements)>;
475     }
476 &smShowAssociationInstances ( "CIM_ElementSoftwareIdentity",
477     $instance.getObjectPath(), #propertylist[] );
478 &smEnd;

```

479 **6.3.2.3 Show Command Form for a Single Object Target – Both References**

480 This command form is for the show verb applied to a single instance. This command form corresponds to
 481 the show command issued against CIM_ElementSoftwareIdentity where both references are specified
 482 and therefore the desired instance is unambiguously identified.

483 6.3.2.3.1 Command Form

```
484 show <CIM_ElementSoftwareIdentity single object>
```

485 6.3.2.3.2 CIM Requirements

486 See CIM_ElementSoftwareIdentity in the “CIM Elements” section of the [Software Inventory Profile](#) for the
487 list of mandatory properties.

488 6.3.2.3.3 Behavior Requirements

489 6.3.2.3.3.1 Preconditions

490 \$instanceA represents the instance of a CIM_ManagedElement and \$instanceB represents the
491 instance of CIM_SoftwareIdentity, both of which are referenced by CIM_ElementSoftwareIdentity.

```
492 $instanceA=<CIM_ManagedElement single object>;  
493 $instanceB=<CIM_SoftwareIdentity single object>;
```

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

495 6.3.2.3.3.2 Pseudo Code

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

505 6.4 CIM_SystemSpecificCollection

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

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

512 **Table 4 – Command Verb Requirements for CIM_SystemSpecificCollection**

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 6.4.2.
start	Not supported	
stop	Not supported	

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

515 6.4.1 Ordering of Results

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

- 518 • Results for CIM_SystemSpecificCollection are unordered; therefore, no algorithm is defined.

519 6.4.2 Show

520 This section describes how to implement the `show` verb when applied to an instance of
521 CIM_SystemSpecificCollection. Implementations shall support the use of the `show` verb with
522 CIM_SystemSpecificCollection.

523 6.4.2.1 Show Command Form for a Single Object Target

524 6.4.2.1.1 Command Form

```
525 show <CIM_SystemSpecificCollection single object>
```

526 6.4.2.1.2 CIM Requirements

527 See CIM_SystemSpecificCollection in the “CIM Elements” section of the [Software Inventory Profile](#) for the
528 list of mandatory properties.

529 6.4.2.1.3 Behavior Requirements

530 6.4.2.1.3.1 Preconditions

531 `$instance` represents the targeted instance of CIM_SystemSpecificCollection.

```
532 $instance=<CIM_SystemSpecificCollection single object>;
```

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

534 6.4.2.1.3.2 Pseudo Code

```
535 #propertylist[] = NULL;
536 if ( false == #all )
537 {
538     #propertylist[] = {<array of mandatory non-key property names (see CIM
539     Requirements)>}
540 }
541 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
542 &smEnd;
```

543 **6.4.2.2 Show Command Form for a Single Object Target – CIM_ComputerSystem or CIM_System**
 544 **Reference**

545 **6.4.2.2.1 Command Form**

546 `show <CIM_SystemSpecificCollection single object>`

547 **6.4.2.2.2 CIM Requirements**

548 See CIM_SystemSpecificCollection in the “CIM Elements” section of the [Software Inventory Profile](#) for the
 549 list of mandatory properties.

550 **6.4.2.2.3 Behavior Requirements**

551 **6.4.2.2.3.1 Preconditions**

552 \$containerInstance contains the instance of CIM_ComputerSystem that is associated with the
 553 targeted instances of CIM_SystemSpecificCollection through the CIM_HostedCollection association.

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

555 **6.4.2.2.3.2 Pseudo Code**

```
556 #propertylist[] = NULL;
557 if ( false == #all )
558 {
559     #propertylist[] = {<array of mandatory non-key property names (see CIM
560         Requirements)>}
561 }
562 &smShowInstances ( "CIM_SystemSpecificCollection", "CIM_HostedCollection",
563     $containerInstance.getObjectPath(), #propertylist[] );
564 &smEnd;
```

565 **6.4.2.3 Show Command Form for Multiple Objects Target - CIM_SoftwareIdentity Reference**

566 **6.4.2.3.1 Command Form**

567 `show <CIM_SystemSpecificCollection multiple objects>`

568 **6.4.2.3.2 CIM Requirements**

569 See CIM_SystemSpecificCollection in the “CIM Elements” section of the [Software Inventory Profile](#) for the
 570 list of mandatory properties.

571 **6.4.2.3.3 Behavior Requirements**

572 **6.4.2.3.3.1 Preconditions**

573 \$containerInstance contains the instance of CIM_SoftwareIdentity that is associated with the
 574 targeted instances of CIM_SystemSpecificCollection through the CIM_MemberOfCollection association.

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

576 **6.4.2.3.3.2 Pseudo Code**

```

577 #propertylist[] = NULL;
578 if ( false == #all )
579     {
580     #propertylist[] = {<array of mandatory non-key property names (see CIM
581     Requirements)>}
582     }
583 &smShowInstances ( "CIM_SystemSpecificCollection", "CIM_MemberOfCollection",
584     $containerInstance.getObjectPath(), #propertylist[] );
585 &smEnd;

```

586 **6.5 CIM_HostedCollection**

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

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

593 **Table 5 – Command Verb Requirements for CIM_HostedCollection**

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	

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

596 **6.5.1 Ordering of Results**

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

- 599 • Results for `CIM_HostedCollection` are unordered; therefore, no algorithm is defined.

600 **6.5.2 Show**

601 This section describes how to implement the `show` verb when applied to an instance of
 602 `CIM_HostedCollection`. Implementations shall support the use of the `show` verb with
 603 `CIM_HostedCollection`.

604 **6.5.2.1 Show Command Form for a Single Object Target – CIM_ComputerSystem or CIM_System** 605 **Reference**

606 This command form is used to show many instances of CIM_HostedCollection. This command form
607 corresponds to a `show` command issued against instances of CIM_HostedCollection where only one
608 reference is specified and the reference is to the scoping instance of CIM_ComputerSystem.

609 **6.5.2.1.1 Command Form**

```
610 show <CIM_HostedCollection single object>
```

611 **6.5.2.1.2 CIM Requirements**

612 See CIM_HostedCollection in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
613 mandatory properties.

614 **6.5.2.1.3 Behavior Requirements**

615 **6.5.2.1.3.1 Preconditions**

616 `$instance` represents the instance of CIM_ComputerSystem or CIM_System, which is referenced by
617 CIM_HostedCollection.

```
618 $instance=<CIM_ComputerSystem single object>;
```

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

620 **6.5.2.1.3.2 Pseudo Code**

```
621 #propertylist[] = NULL;
622 if ( false == #all )
623 {
624     #propertylist[] = <array of mandatory non-key property names (see CIM
625     Requirements)>;
626 }
627 &smShowAssociationInstances ( "CIM_HostedCollection", $instance.getObjectPath(),
628     #propertylist[] );
629 &smEnd;
```

630 **6.5.2.2 Show Command Form for a Single Object – CIM_SystemSpecificCollection Reference**

631 This command form is used to show a single instance of CIM_HostedCollection. This command form
632 corresponds to a `show` command issued against a single instance of CIM_HostedCollection where only
633 one reference is specified and the reference is to the instance of CIM_SystemSpecificCollection.

634 **6.5.2.2.1 Command Form**

```
635 show <CIM_HostedCollection single object>
```

636 **6.5.2.2.2 CIM Requirements**

637 See CIM_HostedCollection in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
638 mandatory properties.

639 **6.5.2.2.3 Behavior Requirements**

640 **6.5.2.2.3.1 Preconditions**

641 `$instance` represents the instance of a CIM_SystemSpecificCollection, which is referenced by
642 CIM_HostedCollection.

```
643 $instance=<CIM_SystemSpecificCollection single object>
```

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

645 6.5.2.2.3.2 Pseudo Code

```
646 #propertylist[] = NULL;
647 if ( false == #all )
648 {
649     #propertylist[] = <array of mandatory non-key property names (see CIM
650     Requirements)>;
651 }
652 &smShowAssociationInstances ( "CIM_HostedCollection", $instance.getObjectPath(),
653     #propertylist[] );
654 &smEnd;
```

655 6.5.2.3 Show Command Form for a Single Object Target – Both References

656 This command form is for the `show` verb applied to a single instance. This command form corresponds to
657 the `show` command issued against `CIM_HostedCollection` where both references are specified and
658 therefore the desired instance is unambiguously identified.

659 6.5.2.3.1 Command Form

```
660 show <CIM_HostedCollection single object>
```

661 6.5.2.3.2 CIM Requirements

662 See `CIM_HostedCollection` in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
663 mandatory properties.

664 6.5.2.3.3 Behavior Requirements

665 6.5.2.3.3.1 Preconditions

666 `$instanceA` represents the instance of `CIM_ComputerSystem` or `CIM_System` and `$instanceB`
667 represents the instance of `CIM_SystemSpecificCollection`, both of which are referenced by
668 `CIM_HostedCollection`.

```
669 $instanceA=<CIM_ComputerSystem single object>;
670 $instanceB=<CIM_SystemSpecificCollection single object>;
```

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

672 6.5.2.3.3.2 Pseudo Code

```
673 #propertylist[] = NULL;
674 if ( false == #all )
675 {
676     #propertylist[] = <array of mandatory non-key property names (see CIM
677     Requirements)>;
678 }
679 &smShowAssociationInstance ( "CIM_HostedCollection", $instanceA.getObjectPath(),
680     $instanceB.getObjectPath(), #propertylist[] );
681 &smEnd;
```


682 6.6 CIM_MemberOfCollection

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

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

689 **Table 6 – 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 6.6.2.
start	Not supported	
stop	Not supported	

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

692 6.6.1 Ordering of Results

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

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

696 6.6.2 Show

697 This section describes how to implement the `show` verb when applied to an instance of
 698 `CIM_MemberOfCollection`. Implementations shall support the use of the `show` verb with
 699 `CIM_MemberOfCollection`.

700 6.6.2.1 Show Command Form for Multiple Objects Target – CIM_SystemSpecificCollection 701 Reference

702 This command form is used to show many instances of `CIM_MemberOfCollection`. This command form
 703 corresponds to a `show` command issued against instances of `CIM_MemberOfCollection` where only one
 704 reference is specified and the reference is to the scoping instance of `CIM_SystemSpecificCollection`.

705 6.6.2.1.1 Command Form

706 `show <CIM_MemberOfCollection multiple objects>`

707 6.6.2.1.2 CIM Requirements

708 See CIM_MemberOfCollection in the “CIM Elements” section of the [Software Inventory Profile](#) for the list
709 of mandatory properties.

710 6.6.2.1.3 Behavior Requirements

711 6.6.2.1.3.1 Preconditions

712 \$instance represents the instance of a CIM_SystemSpecificCollection, which is referenced by
713 CIM_MemberOfCollection.

```
714 $instance=<CIM_SystemSpecificCollection single object>;
```

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

716 6.6.2.1.3.2 Pseudo Code

```
717 #propertylist[] = NULL;
718 if ( false == #all )
719     {
720         #propertylist[] = <array of mandatory non-key property names (see CIM
721             Requirements)>;
722     }
723 &smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.getObjectPath(),
724     #propertylist[] );
725 &smEnd;
```

726 6.6.2.2 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

727 This command form is used to show multiple instances of CIM_MemberOfCollection. This command form
728 corresponds to a show command issued against instances of CIM_MemberOfCollection where only one
729 reference is specified and the reference is to the scoping instance of CIM_SoftwareIdentity.

730 6.6.2.2.1 Command Form

```
731 show <CIM_MemberOfCollection multiple objects>
```

732 6.6.2.2.2 CIM Requirements

733 See CIM_MemberOfCollection in the “CIM Elements” section of the [Software Inventory Profile](#) for the list
734 of mandatory properties.

735 6.6.2.2.3 Behavior Requirements

736 6.6.2.2.3.1 Preconditions

737 \$instance represents the instance of a CIM_SoftwareIdentity, which is referenced by
738 CIM_MemberOfCollection.

```
739 $instance = <CIM_SoftwareIdentity single object>
```

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

741 6.6.2.2.3.2 Pseudo Code

```

742 #propertylist[] = NULL;
743 if ( false == #all )
744 {
745     #propertylist[] = <array of mandatory non-key property names (see CIM
746         Requirements)>;
747 }
748 &smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.getObjectPath(),
749     #propertylist[] );
750 &smEnd;

```

751 6.6.2.3 Show Command Form for a Single Object Target – Both References

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

755 6.6.2.3.1 Command Form

```

756 show <CIM_MemberOfCollection single object>

```

757 6.6.2.3.2 CIM Requirements

758 See `CIM_MemberOfCollection` in the “CIM Elements” section of the [Software Inventory Profile](#) for the list
 759 of mandatory properties.

760 6.6.2.3.3 Behavior Requirements

761 6.6.2.3.3.1 Preconditions

762 `$instanceA` represents the instance of a `CIM_SystemSpecificCollection` and `$instanceB` represents
 763 the instance of `CIM_SoftwareIdentity`, both of which are referenced by `CIM_MemberOfCollection`.

```

764 $instanceA=<CIM_SystemSpecificCollection single object>;
765 $instanceB=<CIM_SoftwareIdentity single object>;

```

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

767 6.6.2.3.3.2 Pseudo Code

```

768 #propertylist[] = NULL;
769 if ( false == #all )
770 {
771     #propertylist[] = <array of mandatory non-key property names (see CIM
772         Requirements)>;
773 }
774 &smShowAssociationInstance ( "CIM_MemberOfCollection", $instanceA.getObjectPath(),
775     $instanceB.getObjectPath(), #propertylist[] );
776 &smEnd;

```

777 6.7 CIM_SoftwareIdentityResource

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

779 Table 7 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 780 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and

781 target. Table 7 is for informational purposes only; in case of a conflict between Table 7 and requirements
 782 detailed in the following sections, the text detailed in the following sections supersedes the information in
 783 Table 7.

784 **Table 7 – Command Verb Requirements for CIM_SoftwareIdentityResource**

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	

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

787 6.7.1 Ordering of Results

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

- 790 • Results for CIM_SoftwareIdentityResource are unordered; therefore, no algorithm is defined.

791 6.7.2 Show

792 This section describes how to implement the `show` verb when applied to an instance of
 793 CIM_SoftwareIdentityResource. Implementations shall support the use of the `show` verb with
 794 CIM_SoftwareIdentityResource.

795 6.7.2.1 Show Command Form for a Single Object Target

796 6.7.2.1.1 Command Form

797 `show <CIM_SoftwareIdentityResource single object>`

798 6.7.2.1.2 CIM Requirements

799 See CIM_SoftwareIdentityResource in the “CIM Elements” section of the [Software Inventory Profile](#) for
 800 the list of mandatory properties.

801 6.7.2.1.3 Behavior Requirements

802 6.7.2.1.3.1 Preconditions

803 `$instance` represents the targeted instance of CIM_SoftwareIdentityResource.

804 `$instance=<CIM_SoftwareIdentityResource single object>;`

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

806 **6.7.2.1.3.2 Pseudo Code**

```

807 #propertylist[] = NULL;
808 if ( false == #all )
809     {
810         #propertylist[] = {<array of mandatory non-key property names (see CIM
811             Requirements)>}
812     }
813 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
814 &smEnd;

```

815 **6.7.2.2 Show Command Form for Multiple Objects Target**816 **6.7.2.2.1 Command Form**

```

817 show <CIM_SoftwareIdentityResource multiple objects>

```

818 **6.7.2.2.2 CIM Requirements**

819 See CIM_SoftwareIdentityResource in the “CIM Elements” section of the [Software Inventory Profile](#) for
820 the list of mandatory properties.

821 **6.7.2.2.3 Behavior Requirements**822 **6.7.2.2.3.1 Preconditions**

823 \$containerInstance contains the instance of CIM_ComputerSystem that is associated with the
824 targeted instances of CIM_SoftwareIdentityResource through the CIM_HostedAccessPoint association.

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

826 **6.7.2.2.3.2 Pseudo Code**

```

827 #propertylist[] = NULL;
828 if ( false == #all )
829     {
830         #propertylist[] = {<array of mandatory non-key property names (see CIM
831             Requirements)>}
832     }
833 &smShowInstances ( "CIM_SoftwareIdentityResource", "CIM_HostedAccessPoint",
834     $containerInstance.getObjectPath(), #propertylist[] );
835 &smEnd;

```

836 **6.8 CIM_SAPAvailableForElement**

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

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

843

Table 8 – Command Verb Requirements for CIM_SAPAvailableForElement

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	

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

846 6.8.1 Ordering of Results

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

- 849 • Results for `CIM_SAPAvailableForElement` are unordered; therefore, no algorithm is defined.

850 6.8.2 Show

851 This section describes how to implement the `show` verb when applied to an instance of
852 `CIM_SAPAvailableForElement`. Implementations shall support the use of the `show` verb with
853 `CIM_SAPAvailableForElement`.

854 6.8.2.1 Show Command Form for a Single Object Target – `CIM_SoftwareIdentity` Reference

855 This command form is used to show a single instance of `CIM_SAPAvailableForElement`. This command
856 form corresponds to a `show` command issued against a single instance of `CIM_SAPAvailableForElement`
857 where only one reference is specified and the reference is to the scoping instance of
858 `CIM_SoftwareIdentity`.

859 6.8.2.1.1 Command Form

```
860 show <CIM_SAPAvailableForElement single object>
```

861 6.8.2.1.2 CIM Requirements

862 See `CIM_SAPAvailableForElement` in the “CIM Elements” section of the [Software Inventory Profile](#) for the
863 list of mandatory properties.

864 6.8.2.1.3 Behavior Requirements

865 6.8.2.1.3.1 Preconditions

866 `instance` represents the instance of a `CIM_SoftwareIdentity`, which is referenced by
867 `CIM_SAPAvailableForElement`.

```
868 $instance=<CIM_SoftwareIdentity single object>
```

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

870 6.8.2.1.3.2 Pseudo Code

```
871 #propertylist[] = NULL;
872 if ( false == #all )
873 {
874     #propertylist[] = <array of mandatory non-key property names (see CIM
875     Requirements)>;
876 }
877 &smShowAssociationInstances ( "CIM_SAPAvailableForElement", $instance.getObjectPath(),
878     #propertylist[] );
879 &smEnd;
```

880 6.8.2.2 Show Command Form for a Single Object Target – CIM_SoftwareIdentityResource 881 Reference

882 This command form is used to show a single instance of CIM_SAPAvailableForElement. This command
883 form corresponds to a `show` command issued against a single instance of CIM_SAPAvailableForElement
884 where only one reference is specified and the reference is to the instance of
885 CIM_SoftwareIdentityResource.

886 6.8.2.2.1 Command Form

```
887 show <CIM_SAPAvailableForElement single object>
```

888 6.8.2.2.2 CIM Requirements

889 See CIM_SAPAvailableForElement in the “CIM Elements” section of the [Software Inventory Profile](#) for the
890 list of mandatory properties.

891 6.8.2.2.3 Behavior Requirements

892 6.8.2.2.3.1 Preconditions

893 \$instance represents the instance of a CIM_SoftwareIdentityResource, which is referenced by
894 CIM_SAPAvailableForElement.

```
895 $instance=<CIM_SoftwareIdentityResource single object>
```

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

897 6.8.2.2.3.2 Pseudo Code

```
898 #propertylist[] = NULL;
899 if ( false == #all )
900 {
901     #propertylist[] = <array of mandatory non-key property names (see CIM
902     Requirements)>;
903 }
904 &smShowAssociationInstances ( "CIM_SAPAvailableForElement", $instance.getObjectPath(),
905     #propertylist[] );
906 &smEnd;
```

907 6.8.2.3 Show Command Form for a Single Object Target – Both References

908 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 909 the `show` command issued against `CIM_SAPAvailableForElement` where both references are specified
 910 and therefore the desired instance is unambiguously identified.

911 6.8.2.3.1 Command Form

```
912 show < CIM_SAPAvailableForElement single object >
```

913 6.8.2.3.2 CIM Requirements

914 See `CIM_SAPAvailableForElement` in the “CIM Elements” section of the [Software Inventory Profile](#) for the
 915 list of mandatory properties.

916 6.8.2.3.3 Behavior Requirements

917 6.8.2.3.3.1 Preconditions

918 `$instanceA` represents the instance of a `CIM_SoftwareIdentity` and `$instanceB` represents the
 919 instance of `CIM_SoftwareIdentityResource`, both of which are referenced by
 920 `CIM_SAPAvailableForElement`.

```
921 $instanceA=<CIM_SoftwareIdentity single object>;  
922 $instanceB=<CIM_SoftwareIdentityResource single object>;
```

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

924 6.8.2.3.3.2 Pseudo Code

```
925 #propertylist[] = NULL;  
926 if ( false == #all )  
927 {  
928     #propertylist[] = <array of mandatory non-key property names (see CIM  
929         Requirements)>;  
930 }  
931 &smShowAssociationInstance ( "CIM_SAPAvailableForElement", $instanceA.getObjectPath(),  
932     $instanceB.getObjectPath(), #propertylist[] );  
933 &smEnd;
```

934 6.9 CIM_HostedAccessPoint

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

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

941 **Table 9 – Command Verb Requirements for `CIM_HostedAccessPoint`**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	

Command Verb	Requirement	Comments
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.9.2.
start	Not supported	
stop	Not supported	

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

944 6.9.1 Ordering of Results

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

- 947 • Results for CIM_HostedAccessPoint are unordered; therefore, no algorithm is defined.

948 6.9.2 Show

949 This section describes how to implement the show verb when applied to an instance of
950 CIM_HostedAccessPoint. Implementations shall support the use of the show verb with
951 CIM_HostedAccessPoint.

952 6.9.2.1 Show Command Form for Multiple Objects Target – CIM_ComputerSystem or 953 CIM_System Reference

954 This command form is used to show many instances of CIM_HostedAccessPoint. This command form
955 corresponds to a show command issued against instances of CIM_HostedAccessPoint where only one
956 reference is specified and the reference is to the scoping instance of CIM_ComputerSystem.

957 6.9.2.1.1 Command Form

```
958 show <CIM_HostedAccessPoint multiple objects>
```

959 6.9.2.1.2 CIM Requirements

960 See CIM_HostedAccessPoint in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
961 mandatory properties.

962 6.9.2.1.3 Behavior Requirements

963 6.9.2.1.3.1 Preconditions

964 \$instance represents the instance of CIM_ComputerSystem or CIM_System, which is referenced by
965 CIM_HostedAccessPoint.

```
966 $instance=<CIM_ComputerSystem single object>;
```

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

968 **6.9.2.1.3.2 Pseudo Code**

```

969 #propertylist[] = NULL;
970 if ( false == #all )
971     {
972     #propertylist[] = <array of mandatory non-key property names (see CIM
973     Requirements)>;
974     }
975 &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath(),
976     #propertylist[] );
977 &smEnd;

```

978 **6.9.2.2 Show Command Form for a Single Object Target – CIM_SoftwareIdentityResource Reference**

980 This command form is used to show a single instance of CIM_HostedAccessPoint. This command form
 981 corresponds to a `show` command issued against an instance of CIM_HostedAccessPoint where only one
 982 reference is specified and the reference is to the instance of CIM_SoftwareIdentityResource.

983 **6.9.2.2.1 Command Form**

```
984 show <CIM_HostedAccessPoint single object>
```

985 **6.9.2.2.2 CIM Requirements**

986 See CIM_HostedAccessPoint in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
 987 mandatory properties.

988 **6.9.2.2.3 Behavior Requirements**989 **6.9.2.2.3.1 Preconditions**

990 \$instance represents the instance of CIM_SoftwareIdentityResource, which is referenced by
 991 CIM_HostedAccessPoint.

```
992 $instance=<CIM_SoftwareIdentityResource single object>;
```

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

994 **6.9.2.2.3.2 Pseudo Code**

```

995 #propertylist[] = NULL;
996 if ( false == #all )
997     {
998     #propertylist[] = <array of mandatory non-key property names (see CIM
999     Requirements)>;
1000     }
1001 &smShowAssociationInstances ( "CIM_HostedAccessPoint", $instance.getObjectPath(),
1002     #propertylist[] );
1003 &smEnd;

```

1004 **6.9.2.3 Show Command Form for a Single Object Target – Both References**

1005 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 1006 the `show` command issued against CIM_HostedAccessPoint where both references are specified and
 1007 therefore the desired instance is unambiguously identified.

1008 **6.9.2.3.1 Command Form**

1009 `show <CIM_HostedAccessPoint single object>`

1010 **6.9.2.3.2 CIM Requirements**

1011 See CIM_HostedAccessPoint in the “CIM Elements” section of the [Software Inventory Profile](#) for the list of
1012 mandatory properties.

1013 **6.9.2.3.3 Behavior Requirements**

1014 \$instanceA represents the instance of a CIM_System or CIM_ComputerSystem and \$instanceB
1015 represents the instance of CIM_SoftwareIdentityResource, both of which are referenced by
1016 CIM_HostedAccessPoint.

1017 `$instanceA=<CIM_ComputerSystem single object>;`
1018 `$instanceB=<CIM_SoftwareIdentityResource single object>;`

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

1020 **6.9.2.3.3.1 Pseudo Code**

```
1021 #propertylist[] = NULL;
1022 if ( false == #all )
1023 {
1024     #propertylist[] = <array of mandatory non-key property names (see CIM
1025         Requirements)>;
1026 }
1027 &smShowAssociationInstance ( "CIM_HostedAccessPoint", $instanceA.getObjectPath(),
1028     $instanceB.getObjectPath(), #propertylist[] );
1029 &smEnd;
```

1030 **6.10 CIM_OrderedComponent**

1031 The cd, help, version, and exit verbs shall be supported as described in [DSP0216](#).

1032 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1033 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1034 target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and
1035 requirements detailed in the following sections, the text detailed in the following sections supersedes the
1036 information in Table 10.

1037 **Table 10 – Command Verb Requirements for CIM_OrderedComponent**

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 6.10.2.
start	Not supported	
stop	Not supported	

1038 No mapping is defined for the following verbs for the specified target: create, delete, dump, load,
1039 reset, start, and stop.

1040 6.10.1 Ordering of Results

1041 When results are returned for multiple instances of CIM_OrderedComponent, implementations shall
1042 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1043 • Results for CIM_OrderedComponent are unordered; therefore, no algorithm is defined.

1044 6.10.2 Show

1045 This section describes how to implement the `show` verb when applied to an instance of
1046 CIM_OrderedComponent. Implementations shall support the use of the `show` verb with
1047 CIM_OrderedComponent.

1048 6.10.2.1 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

1049 This command form is used to show many instances of CIM_OrderedComponent. This command form
1050 corresponds to a `show` command issued against instances of CIM_OrderedComponent where only one
1051 reference is specified and the reference is to the scoping instance of CIM_SoftwareIdentity.

1052 6.10.2.1.1 Command Form

1053 `show <CIM_OrderedComponent multiple objects>`

1054 6.10.2.1.2 CIM Requirements

1055 See CIM_OrderedComponent in the “CIM Elements” section of the [Software Inventory Profile](#) for the list
1056 of mandatory properties.

1057 6.10.2.1.3 Behavior Requirements

1058 6.10.2.1.3.1 Preconditions

1059 `$instance` represents the instance of a CIM_SoftwareIdentity, which is referenced by
1060 CIM_OrderedComponent.

1061 `$instance=<CIM_SoftwareIdentity single object>`

1062 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1063 6.10.2.1.3.2 Pseudo Code

```

1064 #propertylist[] = NULL;
1065 if ( false == #all )
1066     {
1067         #propertylist[] = <array of mandatory non-key property names (see CIM
1068             Requirements)>;
1069     }
1070 &smShowAssociationInstances ( "CIM_OrderedComponent", $instance.getObjectPath(),
1071     #propertylist[] );
1072 &smEnd;

```

1073 6.10.2.2 Show Command Form for a Single Object Target – Both References

1074 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 1075 the `show` command issued against `CIM_OrderedComponent` where both references are specified and
 1076 therefore the desired instance is unambiguously identified.

1077 6.10.2.2.1 Command Form

```

1078 show <CIM_OrderedComponent single object>

```

1079 6.10.2.2.2 CIM Requirements

1080 See `CIM_OrderedComponent` in the “CIM Elements” section of the [Software Inventory Profile](#) for the list
 1081 of mandatory properties.

1082 6.10.2.2.3 Behavior Requirements

1083 `$instanceA` represents the instance of a `CIM_SoftwareIdentity` and `$instanceB` represents the
 1084 instance of `CIM_SoftwareIdentity`, both of which are referenced by `CIM_OrderedComponent`.

```

1085 $instanceA=<CIM_SoftwareIdentity single object>;
1086 $instanceB=<CIM_SoftwareIdentity single object>;

```

1087 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1088 6.10.2.2.3.1 Pseudo Code

```

1089 #propertylist[] = NULL;
1090 if ( false == #all )
1091     {
1092         #propertylist[] = <array of mandatory non-key property names (see CIM
1093             Requirements)>;
1094     }
1095 &smShowAssociationInstance ( "CIM_OrderedComponent", $instanceA.getObjectPath(),
1096     $instanceB.getObjectPath(), #propertylist[] );
1097 &smEnd;

```

1098 6.11 CIM_OrderedDependency

1099 The `cd`, `help`, `version`, and `exit` verbs shall be supported as described in [DSP0216](#).

1100 Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1101 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1102 target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and

1103 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1104 information in Table 11.

1105 **Table 11 – Command Verb Requirements for CIM_OrderedDependency**

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	

1106 No mapping is defined for the following verbs for the specified target: create, delete, dump, exit,
 1107 load, reset, set, start, and stop.

1108 6.11.1 Ordering of Results

1109 When results are returned for multiple instances of CIM_OrderedDependency, implementations shall
 1110 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1111 • Results for CIM_OrderedDependency are unordered; therefore, no algorithm is defined.

1112 6.11.2 Show

1113 This section describes how to implement the `show` verb when applied to an instance of
 1114 CIM_OrderedDependency. Implementations shall support the use of the `show` verb with
 1115 CIM_OrderedDependency.

1116 6.11.2.1 Show Command Form for Multiple Objects Target – CIM_SoftwareIdentity Reference

1117 This command form is used to show many instances of CIM_OrderedDependency. This command form
 1118 corresponds to a `show` command issued against instances of CIM_OrderedDependency where only one
 1119 reference is specified and the reference is to the scoping instance of CIM_SoftwareIdentity.

1120 6.11.2.1.1 Command Form

1121 `show <CIM_OrderedDependency multiple objects>`

1122 6.11.2.1.2 CIM Requirements

1123 See CIM_OrderedDependency in the “CIM Elements” section of the [Software Inventory Profile](#) for the list
 1124 of mandatory properties.

1125 6.11.2.1.3 Behavior Requirements

1126 6.11.2.1.3.1 Preconditions

1127 \$instance represents the instance of a CIM_SoftwareIdentity, which is referenced by
1128 CIM_OrderedDependency.

```
1129 $instance=<CIM_SoftwareIdentity single object>
```

1130 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1131 6.11.2.1.3.2 Pseudo Code

```
1132 #propertylist[] = NULL;
1133 if ( false == #all )
1134 {
1135     #propertylist[] = <array of mandatory non-key property names (see CIM
1136     Requirements)>;
1137 }
1138 &smShowAssociationInstances ( "CIM_OrderedDependency", $instance.getObjectPath(),
1139     #propertylist[] );
1140 &smEnd;
```

1141 6.11.2.2 Show Command Form for a Single Object Target – Both References

1142 This command form is for the show verb applied to a single instance. This command form corresponds to
1143 the show command issued against CIM_OrderedDependency where both references are specified and
1144 therefore the desired instance is unambiguously identified.

1145 6.11.2.2.1 Command Form

```
1146 show <CIM_OrderedDependency single object>
```

1147 6.11.2.2.2 CIM Requirements

1148 See CIM_OrderedDependency in the “CIM Elements” section of the [Software Inventory Profile](#) for the list
1149 of mandatory properties.

1150 6.11.2.2.3 Behavior Requirements

1151 \$instanceA represents the instance of a CIM_SoftwareIdentity and \$instanceB represents the
1152 instance of CIM_SoftwareIdentity, both of which are referenced by CIM_OrderedDependency.

```
1153 $instanceA=<CIM_SoftwareIdentity single object>;
```

```
1154 $instanceB=<CIM_SoftwareIdentity single object>;
```

1155 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1156 6.11.2.2.3.1 Pseudo Code

```
1157 #propertylist[] = NULL;
1158 if ( false == #all )
1159 {
1160     #propertylist[] = <array of mandatory non-key property names (see CIM
1161     Requirements)>;
1162 }
1163 &smShowAssociationInstance ( "CIM_OrderedDependency", $instanceA.getObjectPath(),
1164     $instanceB.getObjectPath(), #propertylist[] );
1165 &smEnd;
```

**ANNEX A
(informative)****Change Log**

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

1166
1167
1168
1169
1170

1171