



1
2
3
4

Document Number: DSP0825

Date: 2009-06-04

Version: 1.0.0

5 **Shared Device Management Profile SM CLP**
6 **Mapping 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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

34

CONTENTS

35	Foreword	5
36	Introduction	6
37	1 Scope	7
38	2 Normative References.....	7
39	2.1 Approved References	7
40	2.2 Other References.....	7
41	3 Terms and Definitions.....	7
42	4 Symbols and Abbreviated Terms.....	8
43	5 Recipes.....	9
44	5.1 IAddReferencedProperties.....	10
45	6 Mappings.....	10
46	6.1 CIM_ElementCapabilities	10
47	6.2 CIM_EnabledLogicalElementCapabilities.....	14
48	6.3 CIM_HostedService	17
49	6.4 CIM_ServiceAffectsElement	19
50	6.5 CIM_SharedDeviceManagementService.....	22
51	6.6 CIM_SharingDependency.....	27
52	6.7 CIM_DeviceSharingCapabilities	30
53	6.8 CIM_LogicalDevice (Sharing Logical Device)	33
54	ANNEX A (informative) Change Log	40
55		

56 Tables

57	Table 1 – Local Recipes.....	9
58	Table 2 – Command Verb Requirements for CIM_ElementCapabilities	11
59	Table 3 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities.....	14
60	Table 4 – Command Verb Requirements for CIM_HostedService	17
61	Table 5 – Command Verb Requirements for CIM_ServiceAffectsElement	20
62	Table 6 – Command Verb Requirements for CIM_SharedDeviceManagementService.....	22
63	Table 7 – Command Verb Requirements for CIM_SharingDependency.....	27
64	Table 8 – Command Verb Requirements for CIM_DeviceSharingCapabilities	30
65	Table 9 – Command Verb Requirements for CIM_LogicalDevice (Sharing Logical Device)	33
66		

68

Foreword

69 The *Shared Device Management Profile SM CLP Mapping Specification* (DSP0825) was prepared by the
70 Server Management Working Group.

71 **Conventions**

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

74 **Acknowledgements**

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

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

85

86

Introduction

87 This document defines the SM CLP mapping for CIM elements described in the [Shared Device](#)
88 [Management Profile](#). The information in this specification, combined with the *SM CLP-to-CIM Common*
89 *Mapping Specification 1.0* ([DSP0216](#)), is intended to be sufficient to implement SM CLP commands
90 relevant to the classes, properties and methods described in the [Shared Device Management Profile](#)
91 using CIM operations.

92 The target audience for this specification is implementers of the SM CLP support for the [Shared Device](#)
93 [Management Profile](#).

94 Shared Device Management Profile SM CLP Mapping 95 Specification

96 1 Scope

97 This specification contains the requirements for an implementation of the SM CLP to provide access to,
98 and implement the behaviors of, the [Shared Device Management Profile](#).

99 2 Normative References

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

103 2.1 Approved References

104 DMTF DSP1021, *Shared Device Management Profile 1.0*,
105 http://www.dmtf.org/standards/published_documents/DSP1021_1.0.pdf

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

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

110 2.2 Other References

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

113 3 Terms and Definitions

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

115 3.1

116 **can**

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

118 3.2

119 **cannot**

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

121 3.3

122 **conditional**

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

- 125 **3.4**
126 **mandatory**
127 indicates requirements to be followed strictly in order to conform to the document and from which no
128 deviation is permitted
- 129 **3.5**
130 **may**
131 indicates a course of action permissible within the limits of the document
- 132 **3.6**
133 **need not**
134 indicates a course of action permissible within the limits of the document
- 135 **3.7**
136 **optional**
137 indicates a course of action permissible within the limits of the document
138
- 139 **3.8**
140 **shall**
141 indicates requirements to be followed strictly in order to conform to the document and from which no
142 deviation is permitted
- 143 **3.9**
144 **shall not**
145 indicates requirements to be followed strictly in order to conform to the document and from which no
146 deviation is permitted
- 147 **3.10**
148 **should**
149 indicates that among several possibilities, one is recommended as particularly suitable, without
150 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 151 **3.11**
152 **should not**
153 indicates that a certain possibility or course of action is deprecated but not prohibited
154

155 **4 Symbols and Abbreviated Terms**

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

- 157 **4.1**
158 **CIM**
159 Common Information Model
- 160 **4.2**
161 **CLP**
162 Command Line Protocol
- 163 **4.3**
164 **DMTF**
165 Distributed Management Task Force

- 166 **4.4**
 167 **IETF**
 168 Internet Engineering Task Force
- 169 **4.5**
 170 **SM**
 171 Server Management
- 172 **4.6**
 173 **SMI-S**
 174 Storage Management Initiative Specification
- 175 **4.7**
 176 **SNIA**
 177 Storage Networking Industry Association

178 **5 Recipes**

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

- 181 • smOpAssociators
- 182 • smOpReferences
- 183 • smOpInvokeMethod
- 184 • smShowAssociationInstances
- 185 • smResetRSC
- 186 • smRequestStateChange
- 187 • smShowInstance
- 188 • &smEnd
- 189 • smProcessError
- 190 • smShowInstances
- 191 • smShowInstanceWithReferenceProperties
- 192 • smShowInstancesWithReferenceProperties

193 For convenience, Table 1 lists each recipe defined in this mapping which is used for more than one verb
 194 or class mapping.

195 **Table 1 – Local Recipes**

Recipe Name	Description	Definition
IAddReferencedProperties	Add associated property to an instance of CIM_LogicalDevice.	See 5.1.

196 The following sections detail Local Recipes defined for use in this mapping.

197 5.1 IAddReferencedProperties

198 5.1.1 Description

199 Add the relevant associated properties to the instance of CIM_LogicalDevice.

200 5.1.2 Preconditions

201 \$device contains the instance of CIM_LogicalDevice to which associated properties should be added.

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

203 5.1.3 Pseudo Code

```
204 sub lAddReferencedProperties($device, #ReferencedPropertyNames) {
205     #propertylist[] = NULL;
206     #Error = &smOpAssociators($device.GetObjectPath(),
207         "CIM_SharingDependency",
208         NULL,
209         NULL,
210         { "CurrentAccess" },
211         $associations);
212     if (0 != #Error.code) {
213         &smProcessOpError (#Error);
214         //includes &smEnd;
215     }
216     //only one allowed to be associated
217     $sharingdependency = $associations[0];
218     $device.CurrentAccess = $sharingDependency.CurrentAccess;
219     #ReferencedPropertyNames = { "CurrentAccess" };
220 } //lAddReferencedProperties()
```

221 6 Mappings

222 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
223 the [Shared Device Management Profile](#).

224 6.1 CIM_ElementCapabilities

225 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

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

231

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 6.1.2.
start	Not supported	
stop	Not supported	

232 No mappings are defined for the following verbs for the specified target: create, delete, dump, load,
233 reset, set, start, and stop.

234 6.1.1 Ordering of Results

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

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

238 6.1.2 Show

239 This section describes how to implement the show verb when applied to an instance of
240 CIM_ElementCapabilities. Implementations shall support the use of the show verb with
241 CIM_ElementCapabilities.

242 The show command is used to display information about the CIM_ElementCapabilities instance or
243 instances.

244 6.1.2.1 Show Multiple Instances – CIM_DeviceSharingCapabilities Reference

245 This command form is for the show verb applied to multiple instances. This command form corresponds
246 to a show command issued against CIM_ElementCapabilities where the reference specified is to an
247 instance of CIM_DeviceSharingCapabilities.

248 6.1.2.1.1 Command Form

```
249 show <CIM_ElementCapabilities single instance>
```

250 6.1.2.1.2 CIM Requirements

251 See CIM_ElementCapabilities in the “CIM Elements” section of the [Shared Device Management Profile](#)
252 for the list of mandatory properties.

253 6.1.2.1.3 Behavior Requirements

254 6.1.2.1.3.1 Preconditions

255 \$instance contains the instance of CIM_DeviceSharingCapabilities which is referenced by
256 CIM_ElementCapabilities.

257 6.1.2.1.3.2 Pseudo Code

```
258 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
259 &smEnd;
```

260 6.1.2.2 Show Multiple Instances – CIM_EnabledLogicalElementCapabilities Reference

261 This command form is for the `show` verb applied to multiple instances. This command form corresponds
262 to a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an
263 instance of `CIM_EnabledLogicalElementCapabilities`.

264 6.1.2.2.1 Command Form

```
265 show <CIM_ElementCapabilities single instance>
```

266 6.1.2.2.2 CIM Requirements

267 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#)
268 for the list of mandatory properties.

269 6.1.2.2.3 Behavior Requirements

270 6.1.2.2.3.1 Preconditions

271 `$instance` contains the instance of `CIM_EnabledLogicalElementCapabilities` which is referenced by
272 `CIM_ElementCapabilities`.

273 6.1.2.2.3.2 Pseudo Code

```
274 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
275 &smEnd;
```

276 6.1.2.3 Show a Single Instance – CIM_LogicalDevice Reference

277 This command form is for the `show` verb applied to a single instance. This command form corresponds to
278 a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an
279 instance of `CIM_LogicalDevice`. An instance of `CIM_LogicalDevice` is referenced by exactly one instance
280 of `CIM_ElementCapabilities`. Therefore, a single instance will be returned.

281 6.1.2.3.1 Command Form

```
282 show <CIM_ElementCapabilities single instance>
```

283 6.1.2.3.2 CIM Requirements

284 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#)
285 for the list of mandatory properties.

286 6.1.2.3.3 Behavior Requirements

287 6.1.2.3.3.1 Preconditions

288 `$instance` contains the instance of `CIM_LogicalDevice` which is referenced by
289 `CIM_ElementCapabilities`.

290 6.1.2.3.3.2 Pseudo Code

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

293 **6.1.2.4 Show a Single Instance – CIM_SharedDeviceManagementService Reference**

294 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 295 a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an
 296 instance of `CIM_SharedDeviceManagementService`. An instance of
 297 `CIM_SharedDeviceManagementService` is referenced by exactly one instance of
 298 `CIM_ElementCapabilities`. Therefore, a single instance will be returned.

299 **6.1.2.4.1 Command Form**

```
300 show <CIM_ElementCapabilities single instance>
```

301 **6.1.2.4.2 CIM Requirements**

302 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#)
 303 for the list of mandatory properties.

304 **6.1.2.4.3 Behavior Requirements**

305 **6.1.2.4.3.1 Preconditions**

306 `$instance` contains the instance of `CIM_SharedDeviceManagementService` which is referenced by
 307 `CIM_ElementCapabilities`.

308 **6.1.2.4.3.2 Pseudo Code**

```
309 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
310 &smEnd;
```

311 **6.1.2.5 Show a Single Instance – Both References (CIM_DeviceSharingCapabilities)**

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

315 **6.1.2.5.1 Command Form**

```
316 show <CIM_ElementCapabilities single instance>
```

317 **6.1.2.5.2 CIM Requirements**

318 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#)
 319 for the list of mandatory properties.

320 **6.1.2.5.3 Behavior Requirements**

321 **6.1.2.5.3.1 Preconditions**

322 `$instanceA` contains the instance of `CIM_DeviceSharingCapabilities` which is referenced by
 323 `CIM_ElementCapabilities`.

324 `$instanceB` contains the instance of `CIM_LogicalDevice` which is referenced by
 325 `CIM_ElementCapabilities`.

326 **6.1.2.5.3.2 Pseudo Code**

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

330 6.1.2.6 Show a Single Instance – Both References (CIM_EnabledLogicalElementCapabilities)

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

334 6.1.2.6.1 Command Form

```
335 show <CIM_ElementCapabilities single instance>
```

336 6.1.2.6.2 CIM Requirements

337 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#)
 338 for the list of mandatory properties.

339 6.1.2.6.3 Behavior Requirements

340 6.1.2.6.3.1 Preconditions

341 `$instanceA` contains the instance of `CIM_EnabledLogicalElementCapabilities` which is referenced by
 342 `CIM_ElementCapabilities`.

343 `$instanceB` contains the instance of `CIM_SharedDeviceManagementService` which is referenced by
 344 `CIM_ElementCapabilities`.

345 6.1.2.6.3.2 Pseudo Code

```
346 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
347     $instanceB.getObjectPath() );
348 &smEnd;
```

349 6.2 CIM_EnabledLogicalElementCapabilities

350 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

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

356 **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	May	See 6.2.2.
show	Shall	See 6.2.3.
start	Not supported	
stop	Not supported	

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

359 6.2.1 Ordering of Results

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

- 362 • Results for CIM_EnabledLogicalElementCapabilities are unordered; therefore, no algorithm is
363 defined.

364 6.2.2 Set

365 This section describes how to implement the `set` verb when applied to an instance of
366 CIM_EnabledLogicalElementCapabilities. Implementations may support the use of the `set` verb with
367 CIM_EnabledLogicalElementCapabilities.

368 6.2.2.1 General Usage of Set for a Single Property

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

371 The requirement for supporting modification of a property using this command form shall be equivalent to
372 the requirement for supporting modification of the property using the ModifyInstance operation as defined
373 in the [Shared Device Management Profile](#).

374 6.2.2.1.1 Command Form

```
375 set <CIM_EnabledLogicalElementCapabilities single instance>  
376 <propertyname>=<propertyvalue>
```

377 6.2.2.1.2 CIM Requirements

378 See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Shared Device
379 Management Profile](#) for the list of mandatory properties.

380 6.2.2.1.3 Behavior Requirements

```
381 $instance=<CIM_EnabledLogicalElementCapabilities single instance>  
382 #propertyName[] = {<propertyname>};  
383 #propertyValues[] = {<propertyvalue>};  
384 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );  
385 &smEnd;
```

386 6.2.2.2 General Usage of Set for Multiple Properties

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

390 The requirement for supporting modification of a property using this command form shall be equivalent to
391 the requirement for supporting modification of the property using the ModifyInstance operation as defined
392 in the [Shared Device Management Profile](#).

393 6.2.2.2.1 Command Form

```
394 set <CIM_EnabledLogicalElementCapabilities single instance>  
395 <propertyname1>=<propertyvalue1><propertynamen>=<propertyvaluen>
```

396 6.2.2.2.2 CIM Requirements

397 See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Shared Device](#)
398 [Management Profile](#) for the list of mandatory properties.

399 6.2.2.2.3 Behavior Requirements

```
400 $instance=<CIM_EnabledLogicalElementCapabilities single instance>
401 #propertyName[] = {<propertyname>};
402 for #i < n
403     {
404         #propertyName[#i] = <propertyname#i>
405         #propertyValue[#i] = <propertyvalue#i>
406     }
407 &smSetInstance ( $instance, #propertyName[], #propertyValue[] );
408 &smEnd;
```

409 6.2.3 Show

410 This section describes how to implement the `show` verb when applied to an instance of
411 CIM_EnabledLogicalElementCapabilities. Implementations shall support the use of the `show` verb with
412 CIM_EnabledLogicalElementCapabilities.

413 The `show` verb is used to display information about an instance or instances of the
414 CIM_EnabledLogicalElementCapabilities class.

415 6.2.3.1 Show a Single Instance

416 This command form is for the `show` verb applied to a single instance of
417 CIM_EnabledLogicalElementCapabilities.

418 6.2.3.1.1 Command Form

```
419 show <CIM_EnabledLogicalElementCapabilities single instance>
```

420 6.2.3.1.2 CIM Requirements

421 See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Shared Device](#)
422 [Management Profile](#) for the list of mandatory properties.

423 6.2.3.1.3 Behavior Requirements

424 6.2.3.1.3.1 Preconditions

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

426 6.2.3.1.3.2 Pseudo Code

```
427 $instance=<CIM_EnabledLogicalElementCapabilities single instance>
428 #propertylist[] = NULL;
429 if ( false == #all )
430     {
431         #propertylist[] = {//all mandatory non-key properties}
432     }
433 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
434 &smEnd;
```


435 **6.2.3.2 Show Multiple Instances**

436 This command form is for the `show` verb applied to multiple instances of
 437 `CIM_EnabledLogicalElementCapabilities`. This command form corresponds to UFS-T-based selection
 438 within a capabilities collection.

439 **6.2.3.2.1 Command Form**

440 `show <CIM_EnabledLogicalElementCapabilities multiple objects>`

441 **6.2.3.2.2 CIM Requirements**

442 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#) for the list of mandatory properties.

444 **6.2.3.2.3 Behavior Requirements**

445 **6.2.3.2.3.1 Preconditions**

446 `$containerInstance` contains the instance of `CIM_ConcreteCollection` for which the contained
 447 `CIM_Capabilities` instances are displayed. `CIM_Capabilities` instances are addressed via an aggregating
 448 instance of `CIM_ConcreteCollection`.

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

450 **6.2.3.2.3.2 Pseudo Code**

```

451 #propertylist[] = NULL;
452 if ( false == #all )
453     {
454         #propertylist[] = { //all mandatory non-key properties};
455     }
456 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
457     $containerInstance.GetObjectPath(), #propertylist[] );
458 &smEnd;
```

459 **6.3 CIM_HostedService**

460 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

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

466 **Table 4 – Command Verb Requirements for CIM_HostedService**

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 6.3.2.
start	Not supported	
stop	Not supported	

467 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
468 `reset`, `set`, `start`, and `stop`.

469 6.3.1 Ordering of Results

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

- 472 • Results for `CIM_HostedService` are unordered; therefore, no algorithm is defined.

473 6.3.2 Show

474 This section describes how to implement the `show` verb when applied to an instance of
475 `CIM_HostedService`. Implementations shall support the use of the `show` verb with `CIM_HostedService`.

476 The `show` command is used to display information about the `CIM_HostedService` instance or instances.

477 6.3.2.1 Show Multiple Instances – `CIM_ComputerSystem` Reference

478 This command form is for the `show` verb applied to multiple instances. This command form corresponds
479 to a `show` command issued against `CIM_HostedService` where only one reference is specified and the
480 reference is to an instance of `CIM_ComputerSystem`.

481 6.3.2.1.1 Command Form

```
482 show <CIM_HostedService multiple objects>
```

483 6.3.2.1.2 CIM Requirements

484 See `CIM_HostedService` in the “CIM Elements” section of the [Shared Device Management Profile](#) for the
485 list of mandatory properties.

486 6.3.2.1.3 Behavior Requirements

487 6.3.2.1.3.1 Preconditions

488 `$instance` contains the instance of `CIM_ComputerSystem` which is referenced by `CIM_HostedService`.

489 6.3.2.1.3.2 Pseudo Code

```
490 &smShowAssociationInstances ( "CIM_HostedService", $instance.GetObjectPath() );  
491 &smEnd;
```

492 6.3.2.2 Show a Single Instance – `CIM_SharedDeviceManagementService` Reference

493 This command form is for the `show` verb applied to a single instance. This command form corresponds to
494 a `show` command issued against `CIM_HostedService` where the reference specified is to an instance of
495 `CIM_SharedDeviceManagementService`. An instance of `CIM_SharedDeviceManagementService` is
496 referenced by exactly one instance of `CIM_HostedService`. Therefore, a single instance will be returned.

497 6.3.2.2.1 Command Form

```
498 show <CIM_HostedService single instance>
```

499 6.3.2.2.2 CIM Requirements

500 See CIM_HostedService in the “CIM Elements” section of the [Shared Device Management Profile](#) for the
501 list of mandatory properties.

502 6.3.2.2.3 Behavior Requirements

503 6.3.2.2.3.1 Preconditions

504 \$instance contains the instance of CIM_SharedDeviceManagementService which is referenced by
505 CIM_HostedService.

506 6.3.2.2.3.2 Pseudo Code

```
507 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );  
508 &smEnd;
```

509 6.3.2.3 Show a Single Instance – Both References

510 This command form is for the `show` verb applied to a single instance. This command form corresponds to
511 a `show` command issued against CIM_HostedService where both references are specified and therefore
512 the desired instance is unambiguously identified.

513 6.3.2.3.1 Command Form

```
514 show <CIM_HostedService single instance>
```

515 6.3.2.3.2 CIM Requirements

516 See CIM_HostedService in the “CIM Elements” section of the [Shared Device Management Profile](#) for the
517 list of mandatory properties.

518 6.3.2.3.3 Behavior Requirements

519 6.3.2.3.3.1 Preconditions

520 \$instanceA contains the instance of CIM_ComputerSystem which is referenced by
521 CIM_HostedService.

522 \$instanceB contains the instance of CIM_SharedDeviceManagementService which is referenced by
523 CIM_HostedService.

524 6.3.2.3.3.2 Pseudo Code

```
525 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),  
526     $instanceB.getObjectPath() );  
527 &smEnd;
```

528 6.4 CIM_ServiceAffectsElement

529 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

530 Table 5 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
531 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
532 target. Table 5 is for informational purposes only; in case of a conflict between Table 5 and requirements

533 detailed in the following sections, the text detailed in the following sections supersedes the information in
 534 Table 5.

535 **Table 5 – Command Verb Requirements for CIM_ServiceAffectsElement**

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	

536 No mappings are defined for the following verbs for the specified target: *create*, *delete*, *dump*, *load*,
 537 *reset*, *set*, *start*, and *stop*.

538 **6.4.1 Ordering of Results**

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

- 541 • Results for CIM_ServiceAffectsElement are unordered; therefore, no algorithm is defined.

542 **6.4.2 Show**

543 This section describes how to implement the *show* verb when applied to an instance of
 544 CIM_ServiceAffectsElement. Implementations shall support the use of the *show* verb with
 545 CIM_ServiceAffectsElement.

546 The *show* command is used to display information about the CIM_ServiceAffectsElement instance or
 547 instances.

548 **6.4.2.1 Show Multiple Instances – CIM_SharedDeviceManagementService Reference**

549 This command form is for the *show* verb applied to multiple instances. This command form corresponds
 550 to a *show* command issued against CIM_ServiceAffectsElement where only one reference is specified
 551 and the reference is to an instance of CIM_SharedDeviceManagementService.

552 **6.4.2.1.1 Command Form**

553 `show <CIM_ServiceAffectsElement multiple objects>`

554 **6.4.2.1.2 CIM Requirements**

555 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Shared Device Management Profile](#)
 556 for the list of mandatory properties.

557 6.4.2.1.3 Behavior Requirements

558 6.4.2.1.3.1 Preconditions

559 \$instance contains the instance of CIM_SharedDeviceManagementService which is referenced by
560 CIM_ServiceAffectsElement.

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

562 6.4.2.1.3.2 Pseudo Code

```
563 #propertylist[] = NULL;
564 if ( false == #all )
565     {
566         #propertylist[] = { //all mandatory non-key properties};
567     }
568 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),
569     #propertylist[] );
570 &smEnd;
```

571 6.4.2.2 Show Multiple Instances – CIM_LogicalDevice Reference

572 This command form is for the show verb applied to a single instance. This command form corresponds to
573 a show command issued against CIM_ServiceAffectsElement where the reference specified is to an
574 instance of CIM_LogicalDevice.

575 6.4.2.2.1 Command Form

```
576 show <CIM_ServiceAffectsElement single instance>
```

577 6.4.2.2.2 CIM Requirements

578 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Shared Device Management Profile](#)
579 for the list of mandatory properties. Section

580 6.4.2.2.3 Behavior Requirements

581 6.4.2.2.3.1 Preconditions

582 \$instance contains the instance of CIM_LogicalDevice which is referenced by
583 CIM_ServiceAffectsElement.

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

585 6.4.2.2.3.2 Pseudo Code

```
586 #propertylist[] = NULL;
587 if ( false == #all )
588     {
589         #propertylist[] = { //all mandatory non-key properties};
590     }
591 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),
592     #propertylist[] );
593 &smEnd;
```

594 6.4.2.3 Show a Single Instance – Both References

595 This command form is for the `show` verb applied to a single instance. This command form corresponds to
596 a `show` command issued against `CIM_ServiceAffectsElement` where both references are specified and
597 therefore the desired instance is unambiguously identified.

598 6.4.2.3.1 Command Form

```
599 show <CIM_ServiceAffectsElement single instance>
```

600 6.4.2.3.2 CIM Requirements

601 See `CIM_ServiceAffectsElement` in the “CIM Elements” section of the [Shared Device Management Profile](#)
602 for the list of mandatory properties.

603 6.4.2.3.3 Behavior Requirements

604 6.4.2.3.3.1 Preconditions

605 `$instanceA` contains the instance of `CIM_ServiceAvailableToElement` which is referenced by
606 `CIM_ServiceAffectsElement`.

607 `$instanceB` contains the instance of `CIM_LogicalDevice` which is referenced by
608 `CIM_ServiceAffectsElement`.

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

610 6.4.2.3.3.2 Pseudo Code

```
611 #propertylist[] = NULL;
612 if ( false == #all )
613 {
614     #propertylist[] = {/all mandatory non-key properties};
615 }
616 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),
617     $instanceB.getObjectPath(), #propertylist[] );
618 &smEnd;
```

619 6.5 CIM_SharedDeviceManagementService

620 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

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

626 **Table 6 – Command Verb Requirements for `CIM_SharedDeviceManagementService`**

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

Command Verb	Requirement	Comments
reset	May	See 6.5.2.
set	May	See 6.5.3.
show	Shall	See 6.5.4.
start	May	See 6.5.5.
stop	May	See 6.5.6.

627 No mappings are defined for the following verbs for the specified target: create, delete, dump, and
628 load.

629 6.5.1 Ordering of Results

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

- 632 • Results for CIM_SharedDeviceManagementService are unordered; therefore, no algorithm is
633 defined.

634 6.5.2 Reset

635 This section describes how to implement the `reset` verb when applied to an instance of
636 CIM_SharedDeviceManagementService. Implementations may support the use of the `reset` verb with
637 CIM_SharedDeviceManagementService.

638 The `reset` verb is used to initiate a reset of the CIM_SharedDeviceManagementService.

639 6.5.2.1 Reset a Single Instance

640 This command form is for the initiation of a reset action against a single instance of
641 CIM_SharedDeviceManagementService. The mapping is implemented as an invocation of the
642 `RequestStateChange()` method on the instance.

643 6.5.2.1.1 Command Form

```
644 reset <CIM_SharedDeviceManagementService single instance>
```

645 6.5.2.1.2 CIM Requirements

```
646 uint16 EnabledState;  
647 uint16 RequestedState;  
648 uint32 EnabledLogicalElement.RequestStateChange (  
649     [IN] uint16 RequestedState,  
650     [OUT] REF CIM_ConcreteJob Job,  
651     [IN] datetime TimeoutPeriod );
```

652 6.5.2.1.3 Behavior Requirements

```
653 $instance=<CIM_SharedDeviceManagementService single instance>  
654 &smResetRSC ( $instance.GetObjectPath() );  
655 &smEnd;
```

656 **6.5.3 Set**

657 This section describes how to implement the `set` verb when applied to an instance of
658 `CIM_SharedDeviceManagementService`. Implementations may support the use of the `set` verb with
659 `CIM_SharedDeviceManagementService`.

660 **6.5.3.1 General Usage of Set for a Single Property**

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

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

666 **6.5.3.1.1 Command Form**

```
667 set <CIM_SharedDeviceManagementService single instance> <propertyname>=<propertyvalue>
```

668 **6.5.3.1.2 CIM Requirements**

669 See `CIM_SharedDeviceManagementService` in the “CIM Elements” section of the [Shared Device
670 Management Profile](#) for the list of mandatory properties.

671 **6.5.3.1.3 Behavior Requirements**

```
672 $instance=<CIM_SharedDeviceManagementService single instance>  
673 #propertyName[] = {<propertyname>};  
674 #propertyValues[] = {<propertyvalue>};  
675 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );  
676 &smEnd;
```

677 **6.5.3.2 General Usage of Set for Multiple Properties**

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

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

684 **6.5.3.2.1 Command Form**

```
685 set <CIM_SharedDeviceManagementService single instance>  
686 <propertyname1>=<propertyvalue1> <propertynamen>=<propertyvaluen>
```

687 **6.5.3.2.2 CIM Requirements**

688 See `CIM_SharedDeviceManagementService` in the “CIM Elements” section of the [Shared Device
689 Management Profile](#) for the list of mandatory properties.

690 6.5.3.2.3 Behavior Requirements

```

691 $instance=<CIM_SharedDeviceManagementService single instance>
692 #propertyName[] = {<propertyname>};
693 for #i < n
694     {
695         #propertyName[#i] = <propertyname#i>
696         #propertyValue[#i] = <propertyvalue#i>
697     }
698 &smSetInstance ( $instance, #propertyName[], #propertyValue[] );
699 &smEnd;

```

700 6.5.4 Show

701 This section describes how to implement the `show` verb when applied to an instance of
 702 `CIM_SharedDeviceManagementService`. Implementations shall support the use of the `show` verb with
 703 `CIM_SharedDeviceManagementService`.

704 The `show` verb is used to display information about the `CIM_SharedDeviceManagementService` instance.

705 6.5.4.1 Show a Single Instance

706 This command form is for the `show` verb applied to a single instance of
 707 `CIM_SharedDeviceManagementService`.

708 6.5.4.1.1 Command Form

```

709 show <CIM_SharedDeviceManagementService single instance>

```

710 6.5.4.1.2 CIM Requirements

711 See `CIM_SharedDeviceManagementService` in the “CIM Elements” section of the [Shared Device
 712 Management Profile](#) for the list of mandatory properties.

713 6.5.4.1.3 Behavior Requirements

714 6.5.4.1.3.1 Preconditions

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

716 6.5.4.1.3.2 Pseudo Code

```

717 #propertylist[] = NULL;
718 if ( false == #all )
719     {
720         #propertylist[] = { //all mandatory non-key properties };
721     }
722 $instance=<CIM_SharedDeviceManagementService single instance>
723 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
724 &smEnd;

```

725 6.5.4.2 Show Multiple Instances

726 This command form is for the `show` verb applied to multiple instances of
 727 `CIM_SharedDeviceManagementService`. This command form corresponds to UFsT-based selection
 728 within a scoping system.

729 **6.5.4.2.1 Command Form**730 `show <CIM_SharedDeviceManagementService multiple objects>`731 **6.5.4.2.2 CIM Requirements**732 See CIM_SharedDeviceManagementService in the “CIM Elements” section of the [Shared Device Management Profile](#) for the list of mandatory properties.
733734 **6.5.4.2.3 Behavior Requirements**735 **6.5.4.2.3.1 Preconditions**736 \$containerInstance contains the instance of CIM_ComputerSystem for which the scoped
737 CIM_SharedDeviceManagementService instances are displayed. The [Shared Device Management Profile](#)
738 requires that the CIM_SharedDeviceManagementService instance be associated with its scoping
739 system via an instance of the CIM_HostedService association.

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

741 **6.5.4.2.3.2 Pseudo Code**742

```
#propertylist[] = NULL;
743 if ( false == #all )
744     {
745         #propertylist[] = { //all mandatory non-key properties};
746     }
747 &smShowInstances ( "CIM_SharedDeviceManagementService", "CIM_HostedService",
748     $containerInstance.getObjectPath(), #propertylist[] );
749 &smEnd;
```

750 **6.5.5 Start**751 This section describes how to implement the `start` verb when applied to an instance of
752 CIM_SharedDeviceManagementService. Implementations may support the use of the `start` verb with
753 CIM_SharedDeviceManagementService.754 The `start` verb is used to enable the device sharing functionality represented by the
755 CIM_SharedDeviceManagementService.756 **6.5.5.1 Start a Single Instance**757 This command form is for the `start` verb applied to a single instance of
758 CIM_SharedDeviceManagementService.759 **6.5.5.1.1 Command Form**760 `start <CIM_SharedDeviceManagementService single instance>`761 **6.5.5.1.2 CIM Requirements**762

```
uint16 EnabledState;
763 uint16 RequestedState;
764 uint32 EnabledLogicalElement.RequestStateChange (
765     [IN] uint16 RequestedState,
766     [OUT] REF CIM_ConcreteJob Job,
767     [IN] datetime TimeoutPeriod );
```

768 **6.5.5.1.3 Behavior Requirements**

```
769 $instance=<CIM_SharedDeviceManagementService single instance>
770 &smStartRSC ( $instance.GetObjectPath() );
771 &smEnd;
```

772 **6.5.6 Stop**

773 This section describes how to implement the `stop` verb when applied to an instance of
 774 `CIM_SharedDeviceManagementService`. Implementations may support the use of the `stop` verb with
 775 `CIM_SharedDeviceManagementService`.

776 The `stop` verb is used to disable the functionality represented by the
 777 `CIM_SharedDeviceManagementService` instance.

778 **6.5.6.1 Stop a Single Instance**

779 This command form is for the `stop` verb applied to a single instance of
 780 `CIM_SharedDeviceManagementService`.

781 **6.5.6.1.1 Command Form**

```
782 stop <CIM_SharedDeviceManagementService single instance>
```

783 **6.5.6.1.2 CIM Requirements**

```
784 uint16 EnabledState;
785 uint16 RequestedState;
786 uint32 EnabledLogicalElement.RequestStateChange (
787     [IN] uint16 RequestedState,
788     [OUT] REF CIM_ConcreteJob Job,
789     [IN] datetime TimeoutPeriod );
```

790 **6.5.6.1.3 Behavior Requirements**

```
791 $instance=<CIM_SharedDeviceManagementService single instance>
792 &smStopRSC ( $instance.GetObjectPath() );
793 &smEnd;
```

794 **6.6 CIM_SharingDependency**

795 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

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

801 **Table 7 – Command Verb Requirements for CIM_SharingDependency**

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 6.6.2.
start	Not supported	
stop	Not supported	

802 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
803 `reset`, `start`, and `stop`.

804 6.6.1 Ordering of Results

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

- 807 • Results for `CIM_SharingDependency` are unordered; therefore, no algorithm is defined.

808 6.6.2 Show

809 This section describes how to implement the `show` verb when applied to an instance of
810 `CIM_SharingDependency`. Implementations shall support the use of the `show` verb with
811 `CIM_SharingDependency`.

812 The `show` command is used to display information about the `CIM_SharingDependency` instance or
813 instances.

814 6.6.2.1 Show Multiple Instances – Real Logical Device

815 This command form is for the `show` verb applied to multiple instances. This command form corresponds
816 to a `show` command issued against `CIM_SharingDependency` where only one reference is specified and
817 the reference is to an instance of `CIM_LogicalDevice` which is a Real Logical Device.

818 6.6.2.1.1 Command Form

```
819 show <CIM_SharingDependency multiple objects>
```

820 6.6.2.1.2 CIM Requirements

821 See `CIM_SharingDependency` in the “CIM Elements” section of the [Shared Device Management Profile](#)
822 for the list of mandatory properties.

823 6.6.2.1.3 Behavior Requirements

824 6.6.2.1.3.1 Preconditions

825 `$instance` contains the instance of `CIM_ComputerSystem` which is referenced by
826 `CIM_SharingDependency`.

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

828 **6.6.2.1.3.2 Pseudo Code**

```

829 #propertylist[] = NULL;
830 if ( false == #all )
831 {
832     #propertylist[] = { //all mandatory non-key properties};
833 }
834 &smShowAssociationInstances ( "CIM_SharingDependency", $instance.getObjectPath(),
835     #propertylist[] );
836 &smEnd;

```

837 **6.6.2.2 Show a Single Instance – Sharing Logical Device Reference**

838 This command form is for the `show` verb applied to a single instance. This command form corresponds to
839 a `show` command issued against `CIM_SharingDependency` where the reference specified is to an
840 instance of `CIM_LogicalDevice` which is a Sharing Logical Device. A Sharing Logical Device is referenced
841 by exactly one instance of `CIM_SharingDependency`. Therefore, a single instance will be returned.

842 **6.6.2.2.1 Command Form**

```
843 show <CIM_SharingDependency single instance>
```

844 **6.6.2.2.2 CIM Requirements**

845 See `CIM_SharingDependency` in the “CIM Elements” section of the [Shared Device Management Profile](#)
846 for the list of mandatory properties.

847 **6.6.2.2.3 Behavior Requirements**848 **6.6.2.2.3.1 Preconditions**

849 `$instance` contains the instance of `CIM_LogicalDevice` which is referenced by
850 `CIM_SharingDependency`.

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

852 **6.6.2.2.3.2 Pseudo Code**

```

853 #propertylist[] = NULL;
854 if ( false == #all )
855 {
856     #propertylist[] = { //all mandatory non-key properties};
857 }
858 &smShowAssociationInstances ( "CIM_SharingDependency", $instance.getObjectPath(),
859     #propertylist[] );
860 &smEnd;

```

861 **6.6.2.3 Show a Single Instance – Both References**

862 This command form is for the `show` verb applied to a single instance. This command form corresponds to
863 a `show` command issued against `CIM_SharingDependency` where both references are specified and
864 therefore the desired instance is unambiguously identified.

865 **6.6.2.3.1 Command Form**

```
866 show <CIM_SharingDependency single instance>
```

867 **6.6.2.3.2 CIM Requirements**

868 See CIM_SharingDependency in the “CIM Elements” section of the [Shared Device Management Profile](#)
869 for the list of mandatory properties.

870 **6.6.2.3.3 Behavior Requirements**871 **6.6.2.3.3.1 Preconditions**

872 \$instanceA contains one of the instances of CIM_LogicalDevice which is referenced by
873 CIM_SharingDependency.

874 \$instanceB contains one of the instances of CIM_LogicalDevice which is referenced by
875 CIM_SharingDependency.

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

877 **6.6.2.3.3.2 Pseudo Code**

```
878 #propertylist[] = NULL;
879 if ( false == #all )
880 {
881     #propertylist[] = {/all mandatory non-key properties};
882 }
883 &smShowAssociationInstance ( "CIM_SharingDependency", $instanceA.getObjectPath(),
884     $instanceB.getObjectPath(), #propertylist[] );
885 &smEnd;
```

886 **6.7 CIM_DeviceSharingCapabilities**

887 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

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

893 **Table 8 – Command Verb Requirements for CIM_DeviceSharingCapabilities**

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	

894 No mappings are defined for the following verbs for the specified target: create, delete, dump, load,
895 reset, set, start, and stop.

896 6.7.1 Ordering of Results

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

- 899 • Results for CIM_DeviceSharingCapabilities are unordered; therefore, no algorithm is defined.

900 6.7.2 Set

901 This section describes how to implement the `set` verb when applied to an instance of
902 CIM_DeviceSharingCapabilities. Implementations may support the use of the `set` verb with
903 CIM_DeviceSharingCapabilities.

904 6.7.2.1 General Usage of Set for a Single Property

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

907 The requirement for supporting modification of a property using this command form shall be equivalent to
908 the requirement for supporting modification of the property using the ModifyInstance operation as defined
909 in the [Shared Device Management Profile](#).

910 6.7.2.1.1 Command Form

```
911 set <CIM_DeviceSharingCapabilities single instance> <propertyname>=<propertyvalue>
```

912 6.7.2.1.2 CIM Requirements

913 See CIM_DeviceSharingCapabilities in the “CIM Elements” section of the [Shared Device Management](#)
914 [Profile](#) for the list of modifiable properties.

915 6.7.2.1.3 Behavior Requirements

```
916 $instance=<CIM_DeviceSharingCapabilities single instance>
917 #propertyName[] = {<propertyname>};
918 #propertyValues[] = {<propertyvalue>};
919 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
920 &smEnd;
```

921 6.7.2.2 General Usage of Set for Multiple Properties

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

925 The requirement for supporting modification of a property using this command form shall be equivalent to
926 the requirement for supporting modification of the property using the ModifyInstance operation as defined
927 in the [Shared Device Management Profile](#).

928 6.7.2.2.1 Command Form

```
929 set <CIM_DeviceSharingCapabilities single instance> <propertyname1>=<propertyvalue1>
930 <propertynamen>=<propertyvaluen>
```

931 6.7.2.2.2 CIM Requirements

932 See CIM_DeviceSharingCapabilities in the “CIM Elements” section of the [Shared Device Management](#)
933 [Profile](#) for the list of mandatory properties.

934 6.7.2.2.3 Behavior Requirements

```

935 $instance=<CIM_DeviceSharingCapabilities single instance>
936 #propertyName[] = {<propertyname>};
937 for #i < n
938     {
939         #propertyName[#i] = <propertyname#i>
940         #propertyValue[#i] = <propertyvalue#i>
941     }
942 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
943 &smEnd;

```

944 6.7.3 Show

945 This section describes how to implement the `show` verb when applied to an instance of
 946 `CIM_DeviceSharingCapabilities`. Implementations shall support the use of the `show` verb with
 947 `CIM_DeviceSharingCapabilities`.

948 The `show` command is used to display information about the `CIM_DeviceSharingCapabilities` instance or
 949 instances.

950 6.7.3.1 Show Multiple Instances – CIM_ConcreteCollection Reference

951 This command form is for the `show` verb applied to multiple instances. This command form corresponds
 952 to a `show` command issued against `CIM_DeviceSharingCapabilities` where only one reference is
 953 specified and the reference is to an instance of `CIM_ConcreteCollection`.

954 6.7.3.1.1 Command Form

```

955 show <CIM_DeviceSharingCapabilities multiple objects>

```

956 6.7.3.1.2 CIM Requirements

957 See `CIM_DeviceSharingCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#)
 958 for the list of mandatory properties.

959 6.7.3.1.3 Behavior Requirements

960 6.7.3.1.3.1 Preconditions

961 `$containerInstance` contains the instance of `CIM_ConcreteCollection` for which we are displaying
 962 scoped `CIM_DeviceSharingCapabilities` instances.

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

964 6.7.3.1.3.2 Pseudo Code

```

965 #propertylist[] = NULL;
966 if ( false == #all )
967     {
968         #propertylist[] = {//all mandatory non-key properties};
969     }
970 &smShowInstances ( "CIM_DeviceSharingCapabilities", "CIM_ElementCapabilities",
971     $containerInstance.getObjectPath(), #propertylist[] );
972 &smEnd;

```


973 **6.7.3.2 Show a Single Instance**

974 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 975 a `show` command issued against a single instance of `CIM_DeviceSharingCapabilities`.

976 **6.7.3.2.1 Command Form**

977 `show <CIM_DeviceSharingCapabilities single instance>`

978 **6.7.3.2.2 CIM Requirements**

979 See `CIM_DeviceSharingCapabilities` in the “CIM Elements” section of the [Shared Device Management Profile](#)
 980 for the list of mandatory properties.

981 **6.7.3.2.3 Behavior Requirements**

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

983 **6.7.3.2.3.1 Pseudo Code**

```

984 #propertylist[] = NULL;
985 if ( false == #all )
986     {
987         #propertylist[] = { //all mandatory non-key properties};
988     }
989 $instance=<CIM_DeviceSharingCapabilities single instance>
990 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
991 &smEnd;
```

992 **6.8 CIM_LogicalDevice (Sharing Logical Device)**

993 This section specifies the command verb requirements for the instance of a subclass of
 994 `CIM_LogicalDevice` which is defined to be a Sharing Logical Device in accordance with the *Shared*
 995 *Device Management Profile*.

996 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

997 Table 9 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 998 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 999 target. Table 9 is for informational purposes only; in case of a conflict between Table 9 and requirements
 1000 detailed in the following sections, the text detailed in the following sections supersedes the information in
 1001 Table 9.

1002 **Table 9 – Command Verb Requirements for CIM_LogicalDevice (Sharing Logical Device)**

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

Command Verb	Requirement	Comments
set	Shall	See 6.8.2.
show	Shall	See 6.8.3.
start	Not supported	
stop	Not supported	

1003 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, and
 1004 `load`.

1005 6.8.1 Ordering of Results

1006 When results are returned for multiple instances of `CIM_SharedDeviceManagementService`,
 1007 implementations shall utilize the following algorithm to produce the natural (that is, default) ordering:

- 1008 • Results for `CIM_SharedDeviceManagementService` are unordered; therefore, no algorithm is
 1009 defined.

1010 6.8.2 Set

1011 This section describes how to implement the `set` verb when applied to an instance of
 1012 `CIM_LogicalDevice`.

1013 6.8.2.1 Using Set to Modify Access to a Shared Device

1014 This command form corresponds to using the `set` verb to assign a value to the `currentaccess` pseudo-
 1015 property of the `CIM_LogicalDevice` instance. This results in the invocation of the `ShareDevice` method on
 1016 an instance of `CIM_SharedDeviceManagementService`.

1017 This command form shall be supported.

1018 6.8.2.1.1 Command Form

```
1019 set <CIM_LogicalDevice single instance> currentaccess=<requestvalue>
```

1020 6.8.2.1.2 CIM Requirements

```
1021 uint16 SharingDependency.CurrentAccess;  

  1022 uint16 SharingDependency.OtherCurrentAccess;  

  1023 uint32 SharedDeviceManagementService.ShareDevice (  

  1024     [IN] uint16 RequestedAccess,  

  1025     [IN] REF CIM_LogicalDevice Device,  

  1026     [IN] REF CIM_System System,  

  1027     [IN] datetime TimeoutPeriod,  

  1028     [IN] Boolean Force,  

  1029     [OUT] REF CIM_ConcreteJob Job);
```

1030 6.8.2.1.3 Behavior Requirements

```
1031 $instance=<CIM_LogicalDevice single instance>  

  1032 //step 1 Find the target System instance  

  1033 #Error = smOpAssociators (  

  1034     $instance.GetObjectPath(),  

  1035     "CIM_SystemDevice",  

  1036     "CIM_System",
```

```

1037     NULL,
1038     NULL,
1039     $Systems[]);
1040 //Only one system is allowed to be associated via SystemDevice
1041 $System-> = $Systems[0].getObjectPath();
1042 //step 2 Find the LogicalDevice instance which reflects the real Logical Device
1043 #Error = smOpAssociators(
1044     $instance.GetObjectPath(),
1045     "CIM_SharingDependency",
1046     "CIM_LogicalDevice",
1047     NULL,
1048     NULL,
1049     $RealDevices[]);
1050 //Only one system is allowed to be associated via SharingDependency
1051 $RealDevice-> = $RealDevices[0].getObjectPath();
1052 // Step 3, find the SharedDeviceManagementService associated to the Real
1053 // Logical Device
1054 #Error = smOpAssociators(
1055     $RealDevice->,
1056     "CIM_ServiceAffectsElement",
1057     "CIM_SharedDeviceManagementService",
1058     NULL,
1059     NULL,
1060     $Services[]);
1061 //Only one instance of SharedDeviceManagementService is allowed to be associated with
1062 //the Real LogicalDevice
1063 $Service-> = $Services[0].getObjectPath();
1064 //Step 4, If Force option was specified, the Force parameter is true, assume the
1065 //option exists and its boolean value is equivalent to whether or not it was
1066 //specified
1067 #ForceParameter = #ForceOption;
1068 //Step 5, Assume a hard-coded default time. Implementations are not required
1069 //to use this specific value.
1070 #TimeoutPeriod = 30;
1071 //Step 6, build parameter lists for method invocation
1072 %InArguments[] = {newArgument("RequestedAccess", <requestvalue>),
1073     newArgument ("Device", #Device),
1074     newArgument ("System", #System),
1075     newArgument("TimeoutPeriod", #TimeoutPeriod),
1076     newArgument ("Force", #ForceParameter)};
1077 %OutArguments[] = { };
1078 //step 7, invoke method
1079 #Error = smOpInvokeMethod ($Service->,
1080     "ShareDevice",
1081     %InArguments[],
1082     %OutArguments[],
1083     #returnStatus);

```

```
1084 //step 8, process return code to CLP Command Status
1085 if (0 != #Error.code) {
1086     //method invocation failed
1087     if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
1088         //if the method invocation contains an embedded error
1089         //use it for the Error for the overall job
1090         &smAddError($job, #Error.$error[0]);
1091         &smMakeCommandStatus($job);
1092         &smEnd;
1093     }
1094     else {
1095         //operation failed, but no detailed error instance, need to make one up
1096         //make an Error instance and associate with job for Operation
1097         $OperationError = smNewInstance("CIM_Error");
1098         //CIM_ERR_FAILED
1099         $OperationError.CIMStatusCode = 1;
1100         //Software Error
1101         $OperationError.ErrorType = 4;
1102         //Unknown
1103         $OperationError.PerceivedSeverity = 0;
1104         $OperationError.OwningEntity = DMTF:SMCLP;
1105         $OperationError.MessageID = 0x00000009;
1106         $OperationError.Message = "An internal software error has occurred.";
1107         &smAddError($job, $OperationError);
1108         &smMakeCommandStatus($job);
1109         &smEnd;
1110     }
1111 } //if CIM op failed
1112 else if (0 == #returnStatus) {
1113     //completed successfully
1114     &smCommandCompleted($job);
1115     &smEnd;
1116 }
1117 else {
1118     //unspecified return code, generic failure
1119     $OperationError = smNewInstance("CIM_Error");
1120     //CIM_ERR_FAILED
1121     $OperationError.CIMStatusCode = 1;
1122     //Other
1123     $OperationError.ErrorType = 1;
1124     //Low
1125     $OperationError.PerceivedSeverity = 2;
1126     $OperationError.OwningEntity = DMTF:SMCLP;
1127     $OperationError.MessageID = 0x00000002;
1128     $OperationError.Message = "Failed. No further information is available.";
1129     &smAddError($job, $OperationError);
1130     &smMakeCommandStatus($job);
1131     &smEnd;
1132 }
```

1133 6.8.2.2 General Usage of Set for a Single Property

1134 This command form corresponds to the general usage of the `set` verb to modify a single property of a
1135 target instance. This is the most common case.

1136 The requirement for supporting modification of a property using this command form shall be equivalent to
1137 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
1138 in the [Shared Device Management Profile](#).

1139 6.8.2.2.1 Command Form

```
1140 set <CIM_LogicalDevice single instance> <propertyname>=<propertyvalue>
```

1141 6.8.2.2.2 CIM Requirements

1142 See `CIM_LogicalDevice` in the “CIM Elements” section of the [Shared Device Management Profile](#) for the
1143 list of modifiable properties.

1144 6.8.2.2.3 Behavior Requirements

```
1145 $instance=<CIM_LogicalDevice single instance>
1146 #propertyName[] = {<propertyname>};
1147 #propertyValues[] = {<propertyvalue>};
1148 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1149 &smEnd;
```

1150 6.8.2.3 General Usage of Set for Multiple Properties

1151 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a
1152 target instance where there is not an explicit relationship between the properties. This is the most
1153 common case.

1154 The requirement for supporting modification of a property using this command form shall be equivalent to
1155 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
1156 in the [Shared Device Management Profile](#).

1157 6.8.2.3.1 Command Form

```
1158 set <CIM_LogicalDevice single instance> <propertyname1>=<propertyvalue1>
1159 <propertynamen>=<propertyvaluen>
```

1160 6.8.2.3.2 CIM Requirements

1161 See `CIM_LogicalDevice` in the “CIM Elements” section of the [Shared Device Management Profile](#) for the
1162 list of mandatory properties.

1163 6.8.2.3.3 Behavior Requirements

```
1164 $instance=<CIM_LogicalDevice single instance>
1165 #propertyName[] = {<propertyname>};
1166 for #i < n
1167 {
1168     #propertyName[#i] = <propertyname#i>
1169     #propertyValues[#i] = <propertyvalue#i>
1170 }
1171 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1172 &smEnd;
```

1173 6.8.3 Show

1174 This section describes how to implement the `show` verb when applied to an instance of
1175 `CIM_LogicalDevice` which is a Sharing Logical Device. Implementations shall support the use of the `show`
1176 verb with `CIM_LogicalDevice`.

1177 The `show` verb is used to display information about the `CIM_LogicalDevice` instance.

1178 6.8.3.1 Show a Single Instance

1179 This command form is for the `show` verb applied to a single instance of `CIM_LogicalDevice`.

1180 6.8.3.1.1 Command Form

```
1181 show <CIM_LogicalDevice single instance>
```

1182 6.8.3.1.2 CIM Requirements

1183 See `CIM_LogicalDevice` in the “CIM Elements” section of the [Shared Device Management Profile](#) for the
1184 list of mandatory properties.

1185 6.8.3.1.3 Behavior Requirements

1186 For the [Shared Device Management Profile](#), a pseudo property is added to the Sharing Logical Device.
1187 The pseudo property has the same name and value as the `CurrentAccess` property on the referencing
1188 `CIM_SharingDependency` instance.

1189 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1190 6.8.3.1.3.1 Pseudo Code

```
1191 #propertylist[] = NULL;
1192 if ( false == #all )
1193     {
1194         #propertylist[] = { "ElementName" }
1195     }
1196 $device=<CIM_LogicalDevice single instance>
1197 &lAddReferencedProperties ( $device, $referencedPropertyNames[], #all );
1198 &smShowInstanceWithReferencedProperties ( $device, #propertyList[],
1199     $referencedPropertyNames[] );
1200 &smEnd;
```

1201 6.8.3.2 Show Multiple Instances

1202 This command form is for the `show` verb applied to multiple instances of `CIM_LogicalDevice`. This
1203 command form corresponds to UFT-based selection within a scoping system.

1204 6.8.3.2.1 Command Form

```
1205 show <CIM_LogicalDevice multiple objects>
```

1206 6.8.3.2.2 CIM Requirements

1207 See `CIM_LogicalDevice` in the “CIM Elements” section of the [Shared Device Management Profile](#) for the
1208 list of mandatory properties.

1209 **6.8.3.2.3 Behavior Requirements**1210 **6.8.3.2.3.1 Preconditions**

1211 \$containerInstance contains the instance of CIM_ComputerSystem for which the scoped
 1212 CIM_LogicalDevice instances are displayed. The [Shared Device Management Profile](#) requires that the
 1213 CIM_LogicalDevice instance be associated with its scoping system via an instance of the
 1214 CIM_SystemDevice association.

1215 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1216 **6.8.3.2.3.2 Pseudo Code**

```

1217 #propertylist[] = NULL;
1218 if ( false == #all )
1219     {
1220     #propertylist[] = { "ElementName" }
1221     }
1222 // Step 1 - find all the scoped instances
1223 #Error = &smOpAssociators (
1224     $containerInstancePath->,
1225     "SystemDevice",
1226     #className,
1227     NULL,
1228     NULL,
1229     NULL,
1230     $devices[] );
1231 if (0 != #Error.code)
1232     {
1233     &smProcessOpError (#Error);
1234     //includes &smEnd;
1235     }
1236 // Step 2 - add their referenced properties
1237 for $device in $devices[] {
1238     &lAddReferencedProperties ( $device, #referencedPropertyNames[] );
1239 }
1240 //step 3 - display them
1241 &smShowInstancesWithReferencedProperties ( $devices[], #propertyList[],
1242     #referencedPropertyNames[] );
1243 &smEnd;

```

1244
1245
1246
1247
1248

ANNEX A
(informative)

Change Log

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

1249