



1
2
3
4

Document Number: DSP0806

Date: 2009-06-04

Version: 1.0.0

5 **Device Tray Profile SM CLP Command Mapping**
6 **Specification**

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

10

11 Copyright notice

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

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

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

33

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 | 6 Mappings..... | 9 |
| 45 | 6.1 CIM_LogicalModule | 9 |
| 46 | 6.2 CIM_SystemDevice | 14 |
| 47 | 6.3 CIM_ConcreteComponent | 16 |
| 48 | 6.4 CIM_ElementCapabilities | 18 |
| 49 | 6.5 CIM_EnabledLogicalElementCapabilities..... | 20 |
| 50 | ANNEX A (informative) Change Log | 23 |
| 51 | | |

52 Tables

| | | |
|----|--|----|
| 53 | Table 1 – Command Verb Requirements for CIM_LogicalModule | 10 |
| 54 | Table 2 – Command Verb Requirements for CIM_SystemDevice | 14 |
| 55 | Table 3 – Command Verb Requirements for CIM_ConcreteComponent | 16 |
| 56 | Table 4 – Command Verb Requirements for CIM_ElementCapabilities | 18 |
| 57 | Table 5 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities..... | 21 |
| 58 | | |

60

Foreword

61 The *Device Tray Profile SM CLP Command Mapping Specification* (DSP0806) was prepared by the
62 Server Management Working Group.

63 **Conventions**

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

66 **Acknowledgements**

67 The authors wish to acknowledge the following participants from the DTMF Server Management Working
68 Group:

- 69 • Aaron Merkin – IBM
- 70 • Jon Hass – Dell
- 71 • Khachatur Papanyan – Dell
- 72 • Jeff Hilland – HP
- 73 • Christina Shaw – HP
- 74 • Jeff Lynch – IBM
- 75 • Perry Vincent – Intel
- 76 • John Leung – Intel

77

78

Introduction

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

83 The target audience for this specification is implementers of the SM CLP support for the [Device Tray](#)
84 [Profile](#).

85 Device Tray Profile SM CLP Command Mapping Specification

86 1 Scope

87 This specification contains the requirements for an implementation of the SM CLP to provide access to,
88 and implement the behaviors of, the [Device Tray Profile](#).

89 2 Normative References

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

93 2.1 Approved References

94 DMTF DSP1019, *DeviceTray Profile 1.0*,
95 http://www.dmtf.org/standards/published_documents/DSP1019_1.0.pdf

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

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

100 2.2 Other References

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

103 3 Terms and Definitions

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

105 3.1

106 **can**

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

108 3.2

109 **cannot**

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

111 3.3

112 **conditional**

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

115 3.4

116 **mandatory**

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

- 119 **3.5**
120 **may**
121 indicates a course of action permissible within the limits of the document
- 122 **3.6**
123 **need not**
124 indicates a course of action permissible within the limits of the document
- 125 **3.7**
126 **optional**
127 indicates a course of action permissible within the limits of the document
- 128 **3.8**
129 **shall**
130 indicates requirements to be followed strictly in order to conform to the document and from which no
131 deviation is permitted
- 132 **3.9**
133 **shall not**
134 indicates requirements to be followed strictly in order to conform to the document and from which no
135 deviation is permitted
- 136 **3.10**
137 **should**
138 indicates that among several possibilities, one is recommended as particularly suitable, without
139 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 140 **3.11**
141 **should not**
142 indicates that a certain possibility or course of action is deprecated but not prohibited

143 **4 Symbols and Abbreviated Terms**

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

- 145 **4.1**
146 **CIM**
147 Common Information Model
- 148 **4.2**
149 **CLP**
150 Command Line Protocol
- 151 **4.3**
152 **DMTF**
153 Distributed Management Task Force
- 154 **4.4**
155 **IETF**
156 Internet Engineering Task Force

157 **4.5**
 158 **SM**
 159 Server Management

160 **4.6**
 161 **SMI-S**
 162 Storage Management Initiative Specification

163 **4.7**
 164 **SNIA**
 165 Storage Networking Industry Association

166 **4.8**
 167 **UFsT**
 168 User Friendly selection Tag

169 **5 Recipes**

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

- 172 • smStartRSC()
- 173 • smStopRSC()
- 174 • smResetRSC()
- 175 • smShowInstance()
- 176 • smShowInstances()
- 177 • smSetInstance()
- 178 • smShowAssociationInstances()
- 179 • smShowAssociationInstance()

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

181 **6 Mappings**

182 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
 183 the [Device Tray Profile](#).

184 **6.1 CIM_LogicalModule**

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

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

191

Table 1 – Command Verb Requirements for CIM_LogicalModule

| Command Verb | Requirement | Comments |
|--------------|---------------|------------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | May | See 6.1.1. |
| set | May | See 6.1.2. |
| show | Shall | See 6.1.3. |
| start | May | See 6.1.4. |
| stop | May | See 6.1.5. |

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

193 6.1.1 Reset

194 This section describes how to implement the `reset` verb when applied to an instance of
195 `CIM_LogicalModule`. Implementations may support the use of the `reset` verb with `CIM_LogicalModule`.

196 The `reset` verb is used to initiate a reset of the `CIM_LogicalModule`.

197 6.1.1.1 Reset a Single Instance

198 This command form is for the initiation of a reset action against a single device tray. The mapping is
199 implemented as an invocation of the `RequestStateChange()` method on the instance.

200 6.1.1.1.1 Command Form

```
201 reset <CIM_LogicalModule single object>
```

202 6.1.1.1.2 CIM Requirements

```
203 uint16 EnabledState;  
204 uint16 RequestedState;  
205 uint32 EnabledLogicalElement.RequestStateChange (  
206     [IN] uint16 RequestedState = "<request value>",  
207     [OUT] REF CIM_ConcreteJob Job,  
208     [IN] datetime TimeoutPeriod );
```

209 6.1.1.1.3 Behavior Requirements

```
210 $instance=<CIM_LogicalModule single object>  
211 smResetRSC ( $instance.GetObjectPath() );  
212 &smEnd;
```

213 6.1.2 Set

214 This section describes how to implement the `set` verb when it is applied to an instance of
215 `CIM_LogicalModule`. Implementations may support the use of the `set` verb with `CIM_LogicalModule`.

216 The `set` verb is used to modify descriptive properties of the `CIM_LogicalModule` instance.

217 6.1.2.1 General Usage of Set for a Single Property

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

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

223 6.1.2.1.1 Command Form

```
224 set <CIM_LogicalModule single instance> <propertyname>=<propertyvalue>
```

225 6.1.2.1.2 CIM Requirements

226 See `CIM_LogicalModule` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of mandatory
227 properties.

228 6.1.2.1.3 Behavior Requirements

```
229 $instance=<CIM_LogicalModule single instance>
230 #propertyName[] = {<propertyname>};
231 #propertyValues[] = {<propertyvalue>};
232 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
233 &smEnd;
```

234 6.1.2.2 General Usage of Set for Multiple Properties

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

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

241 6.1.2.2.1 Command Form

```
242 set <CIM_LogicalModule single instance> <propertyname1>=<propertyvalue1>
243 <propertynamen>=<propertyvaluen>
```

244 6.1.2.2.2 CIM Requirements

245 See `CIM_LogicalModule` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of mandatory
246 properties.

247 6.1.2.2.3 Behavior Requirements

```
248 $instance=<CIM_LogicalModule single instance>
249 #propertyName[] = {<propertyname>};
250 for #i < n
251 {
252     #propertyName[#i] = <propertyname#i>
253     #propertyValues[#i] = <propertyvalue#i>
254 }
255 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
256 &smEnd;
```

257 6.1.3 Show

258 This section describes how to implement the `show` verb when applied to an instance of
259 `CIM_LogicalModule`. Implementations shall support the use of the `show` verb with `CIM_LogicalModule`.

260 The `show` verb is used to display information about the device tray.

261 6.1.3.1 Show a Single Instance

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

263 6.1.3.1.1 Command Form

```
264 show <CIM_LogicalModule single object>
```

265 6.1.3.1.2 CIM Requirements

266 See `CIM_LogicalModule` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of mandatory
267 properties.

268 6.1.3.1.3 Behavior Requirements

269 6.1.3.1.3.1 Preconditions

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

271 6.1.3.1.3.2 Pseudo Code

```
272 $instance=<CIM_LogicalModule single object>  
273 #propertylist[] = NULL;  
274 if (false == #all)  
275 {  
276     #propertylist[] = { "ModuleNumber", "EnabledState", "RequestedState",  
277         "EnabledDefault", "OperationalStatus", "StatusDescriptions",  
278         "LogicalModuleType" }  
279 }  
280 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );  
281 &smEnd;
```

282 6.1.3.2 Show Multiple Instances

283 This command form is for the `show` verb applied to multiple instances of `CIM_LogicalModule`. This
284 command form corresponds to UFsT-based selection within a scoping system.

285 6.1.3.2.1 Command Form

```
286 show <CIM_LogicalModule multiple objects>
```

287 6.1.3.2.2 CIM Requirements

288 See `CIM_LogicalModule` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of mandatory
289 properties.

290 6.1.3.2.3 Behavior Requirements

291 6.1.3.2.3.1 Preconditions

292 \$containerInstance contains the instance of CIM_ComputerSystem for which scoped device trays
 293 (CIM_LogicalModule instances) are being displayed. The [Device Tray Profile](#) requires that the
 294 CIM_LogicalModule instance be associated with its scoping system via an instance of the
 295 CIM_SystemDevice association.

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

297 6.1.3.2.3.2 Pseudo Code

```
298 #propertylist[] = NULL;
299 if (false == #all)
300     {
301         #propertylist[] = { "ModuleNumber", "EnabledState", "RequestedState",
302             "EnabledDefault", "OperationalStatus", "StatusDescriptions",
303             "LogicalModuleType" }
304     }
305 &smShowInstances ( "CIM_LogicalModule", "CIM_SystemDevice",
306     $containerInstance.getObjectPath(), #propertylist[] );
307 &smEnd;
```

308 6.1.4 Start

309 This section describes how to implement the `start` verb when applied to an instance of
 310 CIM_LogicalModule. Implementations may support the use of the `start` verb with CIM_LogicalModule.

311 The `start` verb is used to enable a device tray.

312 6.1.4.1 Start a Single Instance

313 This command form is for the `start` verb applied to a single instance of CIM_LogicalModule.

314 6.1.4.1.1 Command Form

```
315 start <CIM_LogicalModule single object>
```

316 6.1.4.1.2 CIM Requirements

```
317 uint16 EnabledState;
318 uint16 RequestedState;
319 uint32 EnabledLogicalElement.RequestStateChange (
320     [IN] uint16 RequestedState = "<request value>",
321     [OUT] REF CIM_ConcreteJob Job,
322     [IN] datetime TimeoutPeriod );
```

323 6.1.4.1.3 Behavior Requirements

```
324 $instance=<CIM_LogicalModule single object>
325 smStartRSC ( $instance.getObjectPath() );
326 &smEnd;
```

327 **6.1.5 Stop**

328 This section describes how to implement the `stop` verb when applied to an instance of
329 `CIM_LogicalModule`. Implementations may support the use of the `stop` verb with `CIM_LogicalModule`.

330 The `stop` verb is used to disable a device tray.

331 **6.1.5.1 Stop a Single Instance**

332 This command form is for the `stop` verb applied to a single instance of `CIM_LogicalModule`.

333 **6.1.5.1.1 Command Form**

```
334 stop <CIM_LogicalModule single object>
```

335 **6.1.5.1.2 CIM Requirements**

```
336 uint16 EnabledState;
337 uint16 RequestedState;
338 uint32 EnabledLogicalElement.RequestStateChange (
339     [IN] uint16 RequestedState = "<request value>",
340     [OUT] REF CIM_ConcreteJob Job,
341     [IN] datetime TimeoutPeriod );
```

342 **6.1.5.1.3 Behavior Requirements**

```
343 $instance=<CIM_LogicalModule single object>
344 smStopRSC ( $instance.GetObjectPath() );
345 &smEnd;
```

346 **6.2 CIM_SystemDevice**

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

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

353 **Table 2 – Command Verb Requirements for CIM_SystemDevice**

| Command Verb | Requirement | Comments |
|--------------|---------------|------------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |
| set | Not supported | |
| show | Shall | See 6.2.1. |
| start | Not supported | |
| stop | Not supported | |

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

356 **6.2.1 Show**

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

359 The `show` command is used to display information about the `CIM_SystemDevice` instance or instances.

360 **6.2.1.1 Show Multiple Instances**

361 This command form is for the `show` verb applied to multiple instances. This command form corresponds
362 to a `show` command issued against `CIM_SystemDevice` where only one reference is specified and the
363 reference is to an instance of `CIM_ComputerSystem`.

364 **6.2.1.1.1 Command Form**

```
365 show <CIM_SystemDevice multiple objects>
```

366 **6.2.1.1.2 CIM Requirements**

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

369 **6.2.1.1.2.1 Behavior Requirements**

370 **6.2.1.1.2.2 Preconditions**

371 `$instance` contains the instance of `CIM_ComputerSystem` which is referenced by `CIM_SystemDevice`.

372 **6.2.1.1.2.3 Pseudo Code**

```
373 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath() );  
374 &smEnd;
```

375 **6.2.1.2 Show a Single Instance – CIM_LogicalModule Reference**

376 This command form is for the `show` verb applied to a single instance. This command form corresponds to
377 a `show` command issued against `CIM_SystemDevice` where the reference specified is to an instance of
378 `CIM_LogicalModule`. An instance of `CIM_LogicalModule` is referenced by exactly one instance of
379 `CIM_SystemDevice`. Therefore, a single instance will be returned.

380 **6.2.1.2.1 Command Form**

```
381 show <CIM_SystemDevice single object>
```

382 **6.2.1.2.2 CIM Requirements**

383 See `CIM_SystemDevice` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of mandatory
384 properties.

385 **6.2.1.2.3 Behavior Requirements**

386 **6.2.1.2.3.1 Preconditions**

387 `$instance` contains the instance of `CIM_LogicalModule` which is referenced by `CIM_SystemDevice`.

388 **6.2.1.2.3.2 Pseudo Code**

```
389 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath() );  
390 &smEnd;
```

391 **6.2.1.3 Show a Single Instance – Both References**

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

395 **6.2.1.3.1 Command Form**

```
396 show <CIM_SystemDevice single object>
```

397 **6.2.1.3.2 CIM Requirements**

398 See CIM_SystemDevice in the “CIM Elements” section of the [Device Tray Profile](#) for the list of mandatory
 399 properties.

400 **6.2.1.3.3 Behavior Requirements**401 **6.2.1.3.3.1 Preconditions**

402 \$instanceA contains the instance of CIM_ComputerSystem which is referenced by CIM_SystemDevice.

403 \$instanceB contains the instance of CIM_LogicalModule which is referenced by CIM_SystemDevice.

404 **6.2.1.3.3.2 Pseudo Code**

```
405 &smShowAssociationInstance ( "CIM_SystemDevice", $instanceA.getObjectPath(),  

  406     $instanceB.getObjectPath() );  

  407 &smEnd;
```

408 **6.3 CIM_ConcreteComponent**

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

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

415 **Table 3 – Command Verb Requirements for CIM_ConcreteComponent**

| Command Verb | Requirement | Comments |
|--------------|---------------|------------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |
| set | Not supported | |
| show | Shall | See 6.3.1. |
| start | Not supported | |
| stop | Not supported | |

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

418 **6.3.1 Show**

419 This section describes how to implement the `show` verb when applied to an instance of
 420 `CIM_ConcreteComponent`. Implementations shall support the use of the `show` verb with
 421 `CIM_ConcreteComponent`.

422 The `show` command is used to display information about the `CIM_ConcreteComponent` instance or
 423 instances.

424 **6.3.1.1 Show Multiple Instances**

425 This command form is for the `show` verb applied to multiple instances. This command form corresponds
 426 to a `show` command issued against `CIM_ConcreteComponent` where only one reference is specified and
 427 the reference is to an instance of `CIM_LogicalModule`.

428 **6.3.1.1.1 Command Form**

```
429 show <CIM_ConcreteComponent multiple objects>
```

430 **6.3.1.1.2 CIM Requirements**

431 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of
 432 mandatory properties.

433 **6.3.1.1.3 Behavior Requirements**

434 **6.3.1.1.3.1 Preconditions**

435 `$instance` contains the instance of `CIM_LogicalModule` which is referenced by
 436 `CIM_ConcreteComponent`.

437 **6.3.1.1.3.2 Pseudo Code**

```
438 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );  
439 &smEnd;
```

440 **6.3.1.2 Show a Single Instance – CIM_LogicalDevice Reference**

441 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 442 a `show` command issued against `CIM_ConcreteComponent` where the reference specified is to an
 443 instance of `CIM_LogicalDevice`. An instance of `CIM_LogicalDevice` is referenced by exactly one instance
 444 of `CIM_ConcreteComponent`. Therefore, a single instance will be returned.

445 **6.3.1.2.1 Command Form**

```
446 show <CIM_ConcreteComponent single object>
```

447 **6.3.1.2.2 CIM Requirements**

448 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of
 449 mandatory properties.

450 **6.3.1.2.3 Behavior Requirements**

451 **6.3.1.2.3.1 Preconditions**

452 `$instance` contains the instance of `CIM_LogicalDevice` which is referenced by
 453 `CIM_ConcreteComponent`.

454 **6.3.1.2.3.2 Pseudo Code**

```
455 &smShowAssociationInstances ( "CIM_ConcreteComponent", $instance.getObjectPath() );
456 &smEnd;
```

457 **6.3.1.3 Show a Single Instance – Both References**

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

461 **6.3.1.3.1 Command Form**

```
462 show <CIM_ConcreteComponent single object>
```

463 **6.3.1.3.2 CIM Requirements**

464 See `CIM_ConcreteComponent` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of
 465 mandatory properties.

466 **6.3.1.3.3 Behavior Requirements**467 **6.3.1.3.3.1 Preconditions**

468 `$instanceA` contains the instance of `CIM_LogicalDevice` which is referenced by
 469 `CIM_ConcreteComponent`.

470 `$instanceB` contains the instance of `CIM_LogicalModule` which is referenced by
 471 `CIM_ConcreteComponent`.

472 **6.3.1.3.3.2 Pseudo Code**

```
473 &smShowAssociationInstance ( "CIM_ConcreteComponent", $instanceA.getObjectPath(),
474     $instanceB.getObjectPath() );
475 &smEnd;
```

476 **6.4 CIM_ElementCapabilities**

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

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

483 **Table 4 – 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 | |

| Command Verb | Requirement | Comments |
|--------------|---------------|------------|
| show | Shall | See 6.4.1. |
| start | Not supported | |
| stop | Not supported | |

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

486 6.4.1 Show

487 This section describes how to implement the `show` verb when applied to an instance of
488 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with
489 `CIM_ElementCapabilities`.

490 The `show` command is used to display information about the `CIM_ElementCapabilities` instance or
491 instances.

492 6.4.1.1 Show Multiple Instances

493 This command form is for the `show` verb applied to multiple instances. This command form corresponds
494 to a `show` command issued against `CIM_ElementCapabilities` where only one reference is specified and
495 the reference is to an instance of `CIM_LogicalModule`.

496 6.4.1.1.1 Command Form

```
497 show <CIM_ElementCapabilities multiple objects>
```

498 6.4.1.1.2 CIM Requirements

499 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Device Tray Profile](#) for the list of
500 mandatory properties.

501 6.4.1.1.3 Behavior Requirements

502 6.4.1.1.3.1 Preconditions

503 `$instance` contains the instance of `CIM_LogicalModule` which is referenced by
504 `CIM_ElementCapabilities`.

505 6.4.1.1.3.2 Pseudo Code

```
506 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
507 &smEnd;
```

508 6.4.1.2 Show a Single Instance – CIM_LogicalDevice Reference

509 This command form is for the `show` verb applied to a single instance. This command form corresponds to
510 a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an
511 instance of `CIM_LogicalModule`. An instance of `CIM_LogicalModule` is referenced by exactly one instance
512 of `CIM_ElementCapabilities`. Therefore, a single instance will be returned.

513 6.4.1.2.1 Command Form

```
514 show <CIM_ElementCapabilities single object>
```

515 6.4.1.2.2 CIM Requirements

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

518 6.4.1.2.3 Behavior Requirements

519 6.4.1.2.3.1 Preconditions

520 \$instance contains the instance of CIM_LogicalDevice which is referenced by
521 CIM_ElementCapabilities.

522 6.4.1.2.3.2 Pseudo Code

```
523 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
524 &smEnd;
```

525 6.4.1.3 Show a Single Instance – Both References

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

529 6.4.1.3.1 Command Form

```
530 show <CIM_ElementCapabilities single object>
```

531 6.4.1.3.2 CIM Requirements

532 See CIM_ElementCapabilities in the “CIM Elements” section of the [Device Tray Profile](#) for the list of
533 mandatory properties.

534 6.4.1.3.3 Behavior Requirements

535 6.4.1.3.3.1 Preconditions

536 \$instanceA contains the instance of CIM_EnabledLogicalElementCapabilities which is referenced by
537 CIM_ElementCapabilities.

538 \$instanceB contains the instance of CIM_LogicalModule which is referenced by
539 CIM_ElementCapabilities.

540 6.4.1.3.3.2 Pseudo Code

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

544 6.5 CIM_EnabledLogicalElementCapabilities

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

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

551 **Table 5 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities**

| Command Verb | Requirement | Comments |
|--------------|---------------|------------|
| create | Not supported | |
| delete | Not supported | |
| dump | Not supported | |
| load | Not supported | |
| reset | Not supported | |
| set | Not supported | |
| show | Shall | See 6.5.1. |
| start | Not supported | |
| stop | Not supported | |

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

554 **6.5.1 Show**

555 This section describes how to implement the `show` verb when applied to an instance of
556 `CIM_EnabledLogicalElementCapabilities`. Implementations shall support the use of the `show` verb with
557 `CIM_EnabledLogicalElementCapabilities`.

558 The `show` verb is used to display information about the device tray.

559 **6.5.1.1 Show a Single Instance**

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

562 **6.5.1.1.1 Command Form**

```
563 show <CIM_EnabledLogicalElementCapabilities single object>
```

564 **6.5.1.1.2 CIM Requirements**

565 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Device Tray Profile](#) for
566 the list of mandatory properties.

567 **6.5.1.1.3 Behavior Requirements**

568 **6.5.1.1.3.1 Preconditions**

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

570 **6.5.1.1.3.2 Pseudo Code**

```
571 $instance=<CIM_EnabledLogicalElementCapabilities single object>
572 if (#all)
573 {
574     #propertylist[] = { "InstanceID", "RequestedStatesSupported",
575                       "ElementNameEditSupported", "MaxElementNameLen" }
576 }
```

```

577 else
578     {
579         #propertylist[] = { "RequestedStatesSupported", "ElementNameEditSupported",
580             "MaxElementNameLen" }
581     }
582 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
583 &smEnd;

```

584 6.5.1.2 Show Multiple Instances

585 This command form is for the `show` verb applied to multiple instances of
586 `CIM_EnabledLogicalElementCapabilities`. This command form corresponds to UFT-based selection
587 within a capabilities collection.

588 6.5.1.2.1 Command Form

```
589 show <CIM_EnabledLogicalElementCapabilities multiple objects>
```

590 6.5.1.2.2 CIM Requirements

591 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Device Tray Profile](#) for
592 the list of mandatory properties.

593 6.5.1.2.3 Behavior Requirements

594 6.5.1.2.3.1 Preconditions

595 `$containerInstance` contains the instance of `CIM_ConcreteCollection` for which contained
596 `CIM_Capabilities` instances are being displayed. `CIM_Capabilities` instances are addressed via the
597 aggregating instance of `CIM_ConcreteCollection`.

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

599 6.5.1.2.3.2 Pseudo Code

```

600 if (#all)
601     {
602         #propertylist[] = { "InstanceID", "RequestedStatesSupported",
603             "ElementNameEditSupported", "MaxElementNameLen" }
604     }
605 else
606     {
607         #propertylist[] = { "RequestedStatesSupported", "ElementNameEditSupported",
608             "MaxElementNameLen" }
609     }
610 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
611     $containerInstance.getObjectPath(), #propertylist[] );
612 &smEnd;

```

613
614
615
616
617

ANNEX A

(informative)

Change Log

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

618