



1  
2  
3  
4

**Document Number: DSP0807**

**Date: 2009-06-04**

**Version: 1.0.0**

5 **Pass-Through Module Profile SM CLP Command**  
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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party  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

## CONTENTS

34	Foreword .....	5
35	Introduction .....	6
36	1 Scope .....	7
37	2 Normative References.....	7
38	2.1 Approved References .....	7
39	2.2 Other References.....	7
40	3 Terms and Definitions.....	7
41	4 Symbols and Abbreviated Terms.....	8
42	5 Recipes.....	9
43	6 Mappings.....	9
44	6.1 CIM_ElementCapabilities .....	9
45	6.2 CIM_EnabledLogicalElementCapabilities.....	12
46	6.3 CIM_PassThroughModule .....	14
47	6.4 CIM_SystemDevice .....	21
48	ANNEX A (informative) Change Log .....	24
49		

### 50 **Tables**

51	Table 1 – Command Verb Requirements for CIM_ElementCapabilities .....	10
52	Table 2 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities.....	12
53	Table 3 – Command Verb Requirements for CIM_PassThroughModule .....	14
54	Table 4 – Command Verb Requirements for CIM_SystemDevice .....	21
55		



57

## Foreword

58 The *Pass-Through Module Profile SM CLP Command Mapping Specification* (DSP0807) was prepared by  
59 the Server Management Working Group.

### 60 **Conventions**

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

### 63 **Acknowledgements**

64 The authors wish to acknowledge the following participants from the DTMF Server Management Working  
65 Group:

- 66 • Aaron Merkin – IBM
- 67 • Jon Hass – Dell
- 68 • Khachatur Papanyan – Dell
- 69 • Enoch Suen – Dell
- 70 • Jeff Hilland – HP
- 71 • Christina Shaw – HP
- 72 • Perry Vincent – Intel
- 73 • John Leung – Intel

74

75

## Introduction

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

80 The target audience for this specification is implementers of the SM CLP support for the [Pass-Through](#)  
81 [Module Profile](#).

# 82 Pass-Through Module Profile SM CLP Command Mapping 83 Specification

## 84 1 Scope

85 This specification contains the requirements for an implementation of the SM CLP to provide access to,  
86 and implement the behaviors of, the [Pass-Through Module Profile](#).

## 87 2 Normative References

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

### 91 2.1 Approved References

92 DMTF DSP1020, *Pass-Through Module Profile 1.0*,  
93 [http://www.dmtf.org/standards/published\\_documents/DSP1020\\_1.0.pdf](http://www.dmtf.org/standards/published_documents/DSP1020_1.0.pdf)

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

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

### 98 2.2 Other References

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

## 101 3 Terms and Definitions

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

### 103 3.1

#### 104 **can**

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

### 106 3.2

#### 107 **cannot**

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

### 109 3.3

#### 110 **conditional**

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

- 113 **3.4**  
114 **mandatory**  
115 indicates requirements to be followed strictly in order to conform to the document and from which no  
116 deviation is permitted
- 117 **3.5**  
118 **may**  
119 indicates a course of action permissible within the limits of the document
- 120 **3.6**  
121 **need not**  
122 indicates a course of action permissible within the limits of the document
- 123 **3.7**  
124 **optional**  
125 indicates a course of action permissible within the limits of the document
- 126 **3.8**  
127 **shall**  
128 indicates requirements to be followed strictly in order to conform to the document and from which no  
129 deviation is permitted
- 130 **3.9**  
131 **shall not**  
132 indicates requirements to be followed strictly in order to conform to the document and from which no  
133 deviation is permitted
- 134 **3.10**  
135 **should**  
136 indicates that among several possibilities, one is recommended as particularly suitable, without  
137 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 138 **3.11**  
139 **should not**  
140 indicates that a certain possibility or course of action is deprecated but not prohibited

## 141 **4 Symbols and Abbreviated Terms**

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

- 143 **4.1**  
144 **CIM**  
145 Common Information Model
- 146 **4.2**  
147 **CLP**  
148 Command Line Protocol
- 149 **4.3**  
150 **DMTF**  
151 Distributed Management Task Force



- 152 **4.4**  
153 **IETF**  
154 Internet Engineering Task Force
- 155 **4.5**  
156 **SM**  
157 Server Management
- 158 **4.6**  
159 **SMI-S**  
160 Storage Management Initiative Specification
- 161 **4.7**  
162 **SNIA**  
163 Storage Networking Industry Association
- 164 **4.8**  
165 **UFsT**  
166 User Friendly selection Tag

## 167 **5 Recipes**

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

- 170 • smStartRSC
- 171 • smStopRSC
- 172 • smResetRSC
- 173 • smShowInstance
- 174 • smShowInstances
- 175 • smSetInstance
- 176 • smShowAssociationInstances
- 177 • smShowAssociationInstance
- 178 • smMakeCommandStatus
- 179 • smAddError
- 180 • smCommandCompleted

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

## 182 **6 Mappings**

183 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in  
184 the [Pass-Through Module Profile](#).

### 185 **6.1 CIM\_ElementCapabilities**

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

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

192 **Table 1 – 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	

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

### 195 6.1.1 Ordering of Results

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

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

### 199 6.1.2 Show

200 This section describes how to implement the `show` verb when applied to an instance of  
 201 CIM\_ElementCapabilities. Implementations shall support the use of the `show` verb with  
 202 CIM\_ElementCapabilities.

203 The `show` command is used to display information about the CIM\_ElementCapabilities instance or  
 204 instances.

#### 205 6.1.2.1 Show Multiple Instances – CIM\_EnabledLogicalElementCapabilities Reference

206 This command form is for the `show` verb applied to multiple instances. This command form corresponds  
 207 to a `show` command issued against CIM\_ElementCapabilities where only one reference is specified and  
 208 the reference is to an instance of CIM\_EnabledLogicalElementCapabilities.

##### 209 6.1.2.1.1 Command Form

210 `show <CIM_ElementCapabilities multiple instances>`

##### 211 6.1.2.1.2 CIM Requirements

212 See the “CIM Elements” section of the [Pass-Through Module Profile](#).

### 213 6.1.2.1.3 Behavior Requirements

#### 214 6.1.2.1.3.1 Preconditions

215 \$instance contains the instance of CIM\_EnabledLogicalElementCapabilities which is referenced by  
216 CIM\_ElementCapabilities.

#### 217 6.1.2.1.3.2 Pseudo Code

```
218 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
219 &smEnd;
```

### 220 6.1.2.2 Show a Single Instance – CIM\_PassThroughModule Reference

221 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
222 a `show` command issued against `CIM_ElementCapabilities` where the reference specified is to an  
223 instance of `CIM_PassThroughModule`. A single instance of `CIM_EnabledLogicalElementCapabilities` can  
224 be associated with each instance of a `CIM_PassThroughModule`. Therefore, a single instance will be  
225 returned.

#### 226 6.1.2.2.1 Command Form

```
227 show <CIM_ElementCapabilities single instance>
```

#### 228 6.1.2.2.2 CIM Requirements

229 See the “CIM Elements” section of the [Pass-Through Module Profile](#).

### 230 6.1.2.2.3 Behavior Requirements

#### 231 6.1.2.2.3.1 Preconditions

232 \$instance contains the instance of `CIM_PassThroughModule` which is referenced by  
233 `CIM_ElementCapabilities`.

#### 234 6.1.2.2.3.2 Pseudo Code

```
235 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
236 &smEnd;
```

### 237 6.1.2.3 Show a Single Instance – Both References

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

#### 241 6.1.2.3.1 Command Form

```
242 show <CIM_ElementCapabilities single instance>
```

#### 243 6.1.2.3.2 CIM Requirements

244 See the “CIM Elements” section of the [Pass-Through Module Profile](#).

245 **6.1.2.3.3 Behavior Requirements**246 **6.1.2.3.3.1 Preconditions**

247 \$instanceA contains the instance of CIM\_EnabledLogicalElementCapabilities which is referenced by  
248 CIM\_ElementCapabilities.

249 \$instanceB contains the instance of CIM\_PassThroughModule which is referenced by  
250 CIM\_ElementCapabilities.

251 **6.1.2.3.3.2 Pseudo Code**

```
252 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
253     $instanceB.getObjectPath() );
254 &smEnd;
```

255 **6.2 CIM\_EnabledLogicalElementCapabilities**

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

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

262 **Table 2 – 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 6.2.2.
start	Not supported	
stop	Not supported	

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

265 **6.2.1 Ordering of Results**

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

- 268 • Results for CIM\_EnabledLogicalElementCapabilities are unordered; therefore, no algorithm is  
269 defined.

## 270 6.2.2 Show

271 This section describes how to implement the `show` verb when applied to an instance of  
 272 `CIM_EnabledLogicalElementCapabilities`. Implementations shall support the use of the `show` verb with  
 273 `CIM_EnabledLogicalElementCapabilities`.

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

### 276 6.2.2.1 Show a Single Instance

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

#### 279 6.2.2.1.1 Command Form

```
280 show <CIM_EnabledLogicalElementCapabilities single instance>
```

#### 281 6.2.2.1.2 CIM Requirements

282 See the “CIM Elements” section of the [Pass-Through Module Profile](#).

#### 283 6.2.2.1.3 Behavior Requirements

##### 284 6.2.2.1.3.1 Preconditions

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

##### 286 6.2.2.1.3.2 Pseudo Code

```
287 $instance=<CIM_EnabledLogicalElementCapabilities single instance>
288 #propertylist[] = NULL;
289 if ( false == #all) {
290     #propertylist[] = { //all mandatory non-key properties}
291 }
292 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
293 &smEnd;
```

### 294 6.2.2.2 Show Multiple Instances

295 This command form is for the `show` verb applied to multiple instances of  
 296 `CIM_EnabledLogicalElementCapabilities`. This command form corresponds to UFsT-based selection  
 297 within a capabilities collection.

#### 298 6.2.2.2.1 Command Form

```
299 show <CIM_EnabledLogicalElementCapabilities multiple instances>
```

#### 300 6.2.2.2.2 CIM Requirements

301 See the “CIM Elements” section of the [Pass-Through Module Profile](#).

302 **6.2.2.2.3 Behavior Requirements**303 **6.2.2.2.3.1 Preconditions**

304 \$containerInstance contains the instance of CIM\_ConcreteCollection for which we are displaying  
 305 contained CIM\_Capabilities instances CIM\_Capabilities instances are addressed via the aggregating  
 306 instance of CIM\_ConcreteCollection.

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

308 **6.2.2.2.3.2 Pseudo Code**

```
309 #propertylist[] = NULL;
310 if ( false == #all) {
311     #propertylist[] = {/all mandatory non-key properties}
312 }
313 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",
314     $containerInstance.getObjectPath(), #propertylist[] );
315 &smEnd;
```

316 **6.3 CIM\_PassThroughModule**

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

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

323 **Table 3 – Command Verb Requirements for CIM\_PassThroughModule**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	May	See 6.3.2.
set	May	See 6.3.3.
show	Shall	See 6.3.4.
start	May	See 6.3.5.
stop	May	See 6.3.6.

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

325 **6.3.1 Ordering of Results**

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

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

### 329 **6.3.2 Reset**

330 This section describes how to implement the `reset` verb when applied to an instance of  
 331 `CIM_PassThroughModule`. Implementations may support the use of the `reset` verb with  
 332 `CIM_PassThroughModule`.

333 The `reset` verb is used to initiate a reset of the `CIM_PassThroughModule`.

#### 334 **6.3.2.1 Reset a Single Instance**

335 This command form is for the initiation of a reset action against a single pass-through module. The  
 336 mapping is implemented as an invocation of the `RequestStateChange()` method on the instance.

##### 337 **6.3.2.1.1 Command Form**

```
338 reset <CIM_PassThroughModule single instance>
```

##### 339 **6.3.2.1.2 CIM Requirements**

```
340 uint16 EnabledState;
341 uint16 RequestedState;
342 uint32 EnabledLogicalElement.RequestStateChange (
343     [IN] uint16 RequestedState,
344     [OUT] REF CIM_ConcreteJob Job,
345     [IN] datetime TimeoutPeriod );
```

##### 346 **6.3.2.1.3 Behavior Requirements**

```
347 $instance=<CIM_PassThroughModule single instance>
348 smResetRSC ( $instance.GetObjectPath() );
349 &smEnd;
```

### 350 **6.3.3 Set**

351 This section describes how to implement the `set` verb when applied to an instance of  
 352 `CIM_PassThroughModule`. Implementations may support the use of the `set` verb with  
 353 `CIM_PassThroughModule`.

354 No properties of the `CIM_PassThroughModule` instance are writeable via the intrinsic `ModifyInstance`  
 355 operation. Therefore, the only command form specified is for requesting a state change on the instance  
 356 via assignment to the `RequestedState` property.

#### 357 **6.3.3.1 Using Set to Modify Port Assignments**

358 In this command form the `set` verb is used to modify the mapping between the specified internal and  
 359 external ports. Each property name corresponds to a parameter of the `AssignPorts()` method and the  
 360 property value corresponds to the desired value for the parameter.

##### 361 **6.3.3.1.1 Command Form**

```
362 set <CIM_PassThroughModule Single Instance> mapped=<mapped> internalport=<internal
363     port number> externalport=<external port number>
```

##### 364 **6.3.3.1.2 CIM Requirements**

```
365 uint32 AssignPorts(
366     [IN] boolean Mapped,
367     [IN] uint16 InternalPort,
368     [IN] uint16 ExternalPort);
```

369 **6.3.3.1.3 Behavior Requirements**

```

370 //Input parameters to method come straight from command line
371 #mapped=mapped
372 #InternalPort=internalport
373 #ExternalPort=externalport

374 $instance=<CIM_PassThroughModule single instance>
375 %InArguments[] = {newArgument("Mapped", #Mapped),
376                   newArgument("InternalPort", #InternalPort),
377                   newArgument("ExternalPort", #ExternalPort)}
378 %OutArguments[] = { }
379 #Error = InvokeMethod ($instance.GetObjectPath(),
380                       "AssignPorts",
381                       %InArguments[],
382                       %OutArguments[],
383                       #returnStatus);
384 if (0 != #Error.code) {
385     //method invocation failed
386     if ( (null != #Error.$error) && (null != #Error.$error[0]) ) {
387         //if the method invocation contains an embedded error
388         //use it for the Error for the overall job
389         &smAddError($job, #Error.$error[0]);
390         &smMakeCommandStatus($job);
391         &smEnd;
392     }
393     else if ( 17 == #Error.code ) {
394         //17 - CIM_ERR_METHOD_NOT_FOUND
395         // The specified extrinsic method does not exist.
396         $OperationError = smNewInstance("CIM_Error");
397         // CIM_ERR_METHOD_NOT_FOUND
398         $OperationError.CIMStatusCode = 17;
399         //Software Error
400         $OperationError.ErrorType = 10;
401         //Unknown
402         $OperationError.PerceivedSeverity = 0;
403         $OperationError.OwningEntity = DMTF:SMCLP;
404         $OperationError.MessageID = 0x00000001;
405         $OperationError.Message = "Operation is not supported."
406         &smAddError($job, $OperationError);
407         &smMakeCommandStatus($job);
408         &smEnd;
409     }
410     else {
411         //operation failed, but no detailed error instance, need to make one
412         //make an Error instance and associate with job for Operation
413         $OperationError = smNewInstance("CIM_Error");
414         //CIM_ERR_FAILED
415         $OperationError.CIMStatusCode = 1;
416         //Software Error

```



```

417         $OperationError.ErrorType = 4;
418         //Unknown
419         $OperationError.PerceivedSeverity = 0;
420         $OperationError.OwningEntity = DMTF:SMCLP;
421         $OperationError.MessageID = 0x00000009;
422         $OperationError.Message = "An internal software error has occurred.";
423         &smAddError($job, $OperationError);
424         &smMakeCommandStatus($job);
425         &smEnd;
426     }
427 }//if CIM op failed
428 else if (0 == #returnStatus) {
429     //completed successfully
430     &smCommandCompleted($job);
431     &smEnd;
432 }
433 else if (2 == #returnStatus) {
434     //generic failure
435     $OperationError = smNewInstance("CIM_Error");
436     //CIM_ERR_FAILED
437     $OperationError.CIMStatusCode = 1;
438     //Other
439     $OperationError.ErrorType = 1;
440     //Low
441     $OperationError.PerceivedSeverity = 2;
442     $OperationError.OwningEntity = DMTF:SMCLP;
443     $OperationError.MessageID = 0x00000002;
444     $OperationError.Message = "Failed. No further information is available.";
445     &smAddError($job, $OperationError);
446     &smMakeCommandStatus($job);
447 }
448 else {
449     //unspecified return code, generic failure
450     $OperationError = smNewInstance("CIM_Error");
451     //CIM_ERR_FAILED
452     $OperationError.CIMStatusCode = 1;
453     //Other
454     $OperationError.ErrorType = 1;
455     //Low
456     $OperationError.PerceivedSeverity = 2;
457     $OperationError.OwningEntity = DMTF:SMCLP;
458     $OperationError.MessageID = 0x00000002;
459     $OperationError.Message = "Failed. No further information is available.";
460     &smAddError($job, $OperationError);
461     &smMakeCommandStatus($job);
462     &smEnd;
463 }

```

### 464 6.3.3.2 General Usage of Set for a Single Property

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

467 The requirement for supporting modification of a property using this command form shall be equivalent to  
468 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
469 in the [Pass-Through Module Profile](#).

#### 470 6.3.3.2.1 Command Form

```
471 set <CIM_PassThroughModule single instance> <propertyname>=<propertyvalue>
```

#### 472 6.3.3.2.2 CIM Requirements

473 Any modifiable property. See the “CIM Elements” section of the [Pass-Through Module Profile](#).

#### 474 6.3.3.2.3 Behavior Requirements

```
475 $instance=<CIM_PassThroughModule single instance>
476 #propertyNames[] = {<propertyname>};
477 #propertyValues[] = {<propertyvalue>};
478 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
479 &smEnd;
```

### 480 6.3.3.3 General Usage of Set for Multiple Properties

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

484 The requirement for supporting modification of a property using this command form shall be equivalent to  
485 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined  
486 in the [Pass-Through Module Profile](#).

#### 487 6.3.3.3.1 Command Form

```
488 set <CIM_PassThroughModule single instance> <propertyname1>=<propertyvalue1>
489 <propertynamen>=<propertyvaluen>
```

#### 490 6.3.3.3.2 CIM Requirements

491 See supported properties list above.

#### 492 6.3.3.3.3 Behavior Requirements

```
493 $instance=<CIM_PassThroughModule single instance>
494 #propertyNames[] = {<propertyname>};
495 for #i < n
496 {
497     #propertyNames[#i] = <propertyname#i>
498     #propertyValues[#i] = <propertyvalue#i>
499 }
500
501 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
502 &smEnd;
```

### 503 **6.3.4 Show**

504 This section describes how to implement the `show` verb when applied to an instance of  
505 `CIM_PassThroughModule`. Implementations shall support the use of the `show` verb with  
506 `CIM_PassThroughModule`.

507 The `show` verb is used to display information about the pass-through module.

#### 508 **6.3.4.1 Show a Single Instance**

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

##### 510 **6.3.4.1.1 Command Form**

```
511 show <CIM_PassThroughModule single instance>
```

##### 512 **6.3.4.1.2 CIM Requirements**

##### 513 **6.3.4.1.3 Behavior Requirements**

###### 514 **6.3.4.1.3.1 Preconditions**

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

###### 516 **6.3.4.1.3.2 Pseudo Code**

```
517 #propertylist[] = NULL;  
518 if ( false == #all) {  
519     #propertylist[] = { //all mandatory non-key properties };  
520 }  
521 $instance=<CIM_PassThroughModule single instance>  
522 &smShowInstance ( $instance.getObjectPath(),#propertylist[] );  
523 &smEnd;
```

#### 524 **6.3.4.2 Show Multiple Instances**

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

##### 527 **6.3.4.2.1 Command Form**

```
528 show <CIM_PassThroughModule multiple instances>
```

##### 529 **6.3.4.2.2 Behavior Requirements**

###### 530 **6.3.4.2.2.1 Preconditions**

531 `$containerInstance` contains the instance of `CIM_ComputerSystem` for which we are displaying  
532 scoped pass-through modules (`CIM_PassThroughModule` instances). The Pass-through module Profile  
533 requires the `CIM_PassThroughModule` instance be associated with its scoping system via an instance of  
534 the `CIM_SystemDevice` association.

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

#### 536 6.3.4.2.2 Pseudo Code

```
537 #propertylist[] = NULL;
538 if ( false == #all) {
539     #propertylist[] = { //all mandatory non-key properties };
540 }
541 &smShowInstances ( "CIM_PassThroughModule", "CIM_SystemDevice",
542     $containerInstance.getObjectPath(), #propertylist[] );
543 &smEnd;
```

#### 544 6.3.5 Start

545 This section describes how to implement the `start` verb when applied to an instance of  
546 `CIM_PassThroughModule`. Implementations may support the use of the `start` verb with  
547 `CIM_PassThroughModule`.

548 The `start` verb is used to enable a pass-through module.

##### 549 6.3.5.1 Start a Single Instance

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

###### 551 6.3.5.1.1 Command Form

```
552 start <CIM_PassThroughModule single instance>
```

###### 553 6.3.5.1.2 CIM Requirements

```
554 uint16 EnabledState;
555 uint16 RequestedState;
556 uint32 EnabledLogicalElement.RequestStateChange (
557     [IN] uint16 RequestedState,
558     [OUT] REF CIM_ConcreteJob Job,
559     [IN] datetime TimeoutPeriod );
```

###### 560 6.3.5.1.3 Behavior Requirements

```
561 $instance=<CIM_PassThroughModule single instance>
562 smStartRSC ( $instance.getObjectPath() );
563 &smEnd;
```

#### 564 6.3.6 Stop

565 This section describes how to implement the `stop` verb when applied to an instance of  
566 `CIM_PassThroughModule`. Implementations may support the use of the `stop` verb with  
567 `CIM_PassThroughModule`.

568 The `stop` verb is used to disable a pass-through module.

##### 569 6.3.6.1 Stop a Single Instance

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

###### 571 6.3.6.1.1 Command Form

```
572 stop <CIM_PassThroughModule single instance>
```

573 **6.3.6.1.2 CIM Requirements**

```
574 uint16 EnabledState;
575 uint16 RequestedState;
576 uint32 EnabledLogicalElement.RequestStateChange (
577     [IN] uint16 RequestedState,
578     [OUT] REF CIM_ConcreteJob Job,
579     [IN] datetime TimeoutPeriod );
```

580 **6.3.6.1.3 Behavior Requirements**

```
581 $instance=<CIM_PassThroughModule single instance>
582 smStopRSC ( $instance.getObjectPath() );
583 &smEnd;
```

584 **6.4 CIM\_SystemDevice**

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

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

591 **Table 4 – Command Verb Requirements for CIM\_SystemDevice**

Command Verb	Requirement	Comments
create	Shall not	
delete	Shall not	
dump	Shall not	
load	Shall not	
reset	Shall not	
set	Shall not	
show	Shall	See 6.4.1.
start	Shall not	
stop	Shall not	

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

594 **6.4.1 Show**

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

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

598 **6.4.1.1 Show Multiple Instances – CIM\_ComputerSystem Reference**

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

**602 6.4.1.1.1 Command Form**

```
603 show <CIM_SystemDevice multiple instances>
```

**604 6.4.1.1.2 CIM Requirements****605 6.4.1.1.3 Behavior Requirements****606 6.4.1.1.3.1 Preconditions**

607 \$instance contains the instance of CIM\_ComputerSystem which is referenced by CIM\_SystemDevice.

**608 6.4.1.1.3.2 Pseudo Code**

```
609 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath() );  
610 &smEnd;
```

**611 6.4.1.2 Show a Single Instance – CIM\_PassThroughModule Reference**

612 This command form is for the `show` verb applied to a single instance. This command form corresponds to  
613 a `show` command issued against `CIM_SystemDevice` where the reference specified is to an instance of  
614 `CIM_PassThroughModule`. An instance of `CIM_PassThroughModule` is referenced by exactly one  
615 instance of `CIM_SystemDevice`. Therefore, a single instance will be returned.

**616 6.4.1.2.1 Command Form**

```
617 show <CIM_SystemDevice single instance>
```

**618 6.4.1.2.2 CIM Requirements****619 6.4.1.2.3 Behavior Requirements****620 6.4.1.2.3.1 Preconditions**

621 \$instance contains the instance of `CIM_PassThroughModule` which is referenced by  
622 `CIM_SystemDevice`.

**623 6.4.1.2.3.2 Pseudo Code**

```
624 &smShowAssociationInstances ( "CIM_SystemDevice", $instance.getObjectPath() );  
625 &smEnd;
```

**626 6.4.1.3 Show a Single Instance – Both References**

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

**630 6.4.1.3.1 Command Form**

```
631 show <CIM_SystemDevice single instance>
```

632 **6.4.1.3.2 CIM Requirements**633 **6.4.1.3.3 Behavior Requirements**634 **6.4.1.3.3.1 Preconditions**

635 \$instanceA contains the instance of CIM\_ComputerSystem which is referenced by CIM\_SystemDevice.

636 \$instanceB contains the instance of CIM\_PassThroughModule which is referenced by  
637 CIM\_SystemDevice.

638 **6.4.1.3.3.2 Pseudo Code**

```
639 &smShowAssociationInstance ( "CIM_SystemDevice", $instanceA.getObjectPath(),  
640     $instanceB.getObjectPath() );  
641 &smEnd;
```

642

643  
644  
645  
646  
647

**ANNEX A**  
(informative)

**Change Log**

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

648