



1
2
3
4

Document Number: DSP0824

Date: 2009-06-04

Version: 1.0.0

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

34

CONTENTS

35 Foreword 5
 36 Introduction 6
 37 1 Scope 7
 38 2 Normative References..... 7
 39 2.1 Approved References 7
 40 2.1 Other References..... 7
 41 3 Terms and Definitions..... 7
 42 4 Symbols and Abbreviated Terms..... 8
 43 5 Recipes..... 9
 44 5.1 IAddReferencedProperties..... 9
 45 6 Mappings..... 11
 46 6.1 CIM_ComputerSystem..... 11
 47 6.2 CIM_ElementCapabilities 21
 48 6.3 CIM_EnabledLogicalElementCapabilities..... 23
 49 6.4 CIM_HostedService 25
 50 6.5 CIM_IsSpare 27
 51 6.6 CIM_MemberOfCollection 30
 52 6.7 CIM_OwningCollectionElement 32
 53 6.8 CIM_RedundancySet..... 34
 54 6.9 CIM_ServiceAffectsElement 40
 55 6.10 CIM_TimeService 43
 56 ANNEX A (informative) Change Log 45
 57

58 Tables

59 Table 1 – Local Recipes..... 9
 60 Table 2 – Command Verb Requirements for CIM_ComputerSystem 12
 61 Table 3 – Command Verb Requirements for CIM_ElementCapabilities 21
 62 Table 4 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities..... 23
 63 Table 5 – Command Verb Requirements for CIM_HostedService 25
 64 Table 6 – Command Verb Requirements for CIM_IsSpare 27
 65 Table 7 – Command Verb Requirements for CIM_MemberOfCollection 30
 66 Table 8 – Command Verb Requirements for CIM_OwningCollectionElement 32
 67 Table 9 – Command Verb Requirements for CIM_RedundancySet..... 35
 68 Table 10 – Command Verb Requirements for CIM_ServiceAffectsElement 40
 69 Table 11 – Command Verb Requirements for CIM_TimeService 43
 70

72

Foreword

73 The *Service Processor Profile SM CLP Command Mapping Specification* (DSP0824) was prepared by the
74 Server Management Working Group.

75 **Conventions**

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

78 **Acknowledgements**

79 The authors wish to acknowledge the following participants from the DMTF Server Management Working
80 Group:

- 81 • Aaron Merkin – IBM
- 82 • Jon Hass – Dell
- 83 • Khachatur Papanyan – Dell
- 84 • Jeff Hilland – HP
- 85 • Christina Shaw – HP
- 86 • Perry Vincent – Intel
- 87 • John Leung – Intel

88

89

Introduction

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

94 The target audience for this specification is implementers of the SM CLP support for the [Service](#)
95 [Processor Profile](#).

96
97

Service Processor Profile SM CLP Command Mapping Specification

1 Scope

99 This specification contains the requirements for an implementation of the SM CLP to provide access to,
100 and implement the behaviors of, the [Service Processor Profile](#).

2 Normative References

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

2.1 Approved References

106 DMTF DSP1018, *Service Processor Profile 1.0*,
107 http://www.dmtf.org/standards/published_documents/DSP1018_1.0.pdf

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

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

2.1 Other References

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

3 Terms and Definitions

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

3.1

can

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

3.2

cannot

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

3.3

conditional

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

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

156 **4 Symbols and Abbreviated Terms**

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

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

- 167 **4.4**
 168 **SM**
 169 Server Management
- 170 **4.5**
 171 **SMI-S**
 172 Storage Management Initiative Specification
- 173 **4.6**
 174 **SNIA**
 175 Storage Networking Industry Association
- 176 **4.7**
 177 **UFsT**
 178 User Friendly Selection Tag

179 **5 Recipes**

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

- 182 • smResetRSC
- 183 • smShowInstance
- 184 • smShowInstances
- 185 • smShowAssociationInstance
- 186 • smShowAssociationInstances
- 187 • smStartRSC
- 188 • smStopRSC

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

191 **Table 1 – Local Recipes**

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

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

193 **5.1 IAddReferencedProperties**

194 **5.1.1 Description**

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

196 **5.1.2 Preconditions**

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

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

199 **5.1.3 Pseudo Code**

```

200 sub lAddReferencedProperties($instance, #ReferencedPropertyNames[]){
201     #propertylist[] = NULL;
202     //find the associated service, call the method as a query, insert the time as a
203     referenced property
204     // Find the TimeService associated to the target system
205     #Error = smOpAssociators(
206         $instance.getObjectPath(),
207         "CIM_ServiceAffectsElement",
208         "CIM_TimeService",
209         NULL,
210         NULL,
211         $Services[]);
212     //if there is not a service associated the function is not supported.
213     if (0 == $Services[].length){
214         //not supported, don't add property
215         return;
216     }
217     $Service-> = $Services[0].getObjectPath();
218     %InArguments[] = { newArgument("GetRequest", FALSE)
219                       newArgument("ManagedElement", $instance.getObjectPath())}
220     %OutArguments[] = { newArgument("TimeData", #timedata)};
221     //invoke method
222     #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
223         "ManageTime",
224         %InArguments[],
225         %OutArguments[]);
226
227     // process return code to CLP Command Status
228     if (0 != #Error.code) {
229         //method invocation failed
230         if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
231             // if the method invocation contains an embedded error
232             // use it for the Error for the overall job
233             &smAddError($job, #Error.$error[0]);
234             &smMakeCommandStatus($job);
235             &smEnd;
236         }
237         else if (#Error.code == 17) {
238             //not supported, so don't add property
239             return;
240         }
241         else {
242             //operation failed, but no detailed error instance, need to make one up
243             //make an Error instance and associate with job for Operation
244             $OperationError = smNewInstance("CIM_Error");
245             //CIM_ERR_FAILED
246             $OperationError.CIMStatusCode = 1;
247             //Software Error

```

```

248     $OperationError.ErrorType = 4;
249     //Unknown
250     $OperationError.PerceivedSeverity = 0;
251     $OperationError.OwningEntity = DMTF:SMCLP;
252     $OperationError.MessageID = 0x00000009;
253     $OperationError.Message = "An internal software error has occurred.";
254     &smAddError($job, $OperationError);
255     &smMakeCommandStatus($job);
256     &smEnd;
257 }
258 }//if CIM op failed
259 else if (0 == #returnStatus) {
260     //completed successfully
261     $instance.CurrentTime = #timedata;
262     #ReferencedPropertyNames = {"CurrentTime"};
263 }
264 else if (1 == #returnStatus) {
265     //not supported, so don't add
266     return;
267 }
268 else {
269     //generic failure
270     $OperationError = smNewInstance("CIM_Error");
271     //CIM_ERR_FAILED
272     $OperationError.CIMStatusCode = 1;
273     //Other
274     $OperationError.ErrorType = 1;
275     //Low
276     $OperationError.PerceivedSeverity = 2;
277     $OperationError.OwningEntity = DMTF:SMCLP;
278     $OperationError.MessageID = 0x00000002;
279     $OperationError.Message = "Failed. No further information is available.";
280     &smAddError($job, $OperationError);
281     &smMakeCommandStatus($job);
282 }
283 } //lAddReferencedProperties()

```

284 6 Mappings

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

288 6.1 CIM_ComputerSystem

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

290 Table 2 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 291 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 292 target. Table 2 is for informational purposes only; in case of a conflict between Table 2 and requirements

293 detailed in the following sections, the text detailed in the following sections supersedes the information in
 294 Table 2.

295 **Table 2 – Command Verb Requirements for CIM_ComputerSystem**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.1.2.
Set	May	See 6.1.3.
Show	Shall	See 6.1.4.
Start	May	See 6.1.5.
Stop	May	See 6.1.6.

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

297 6.1.1 Ordering of Results

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

- 300 • Results for CIM_ComputerSystem are unordered; therefore, no algorithm is defined.

301 6.1.2 Reset

302 This section describes how to implement the `reset` verb when applied to an instance of
 303 CIM_ComputerSystem. Implementations may support the use of the `reset` verb with
 304 CIM_ComputerSystem.

305 6.1.2.1 Command Form

```
306 reset <CIM_ComputerSystem single instance>
```

307 6.1.2.2 CIM Requirements

```
308 uint16 EnabledState;  

  309 uint16 RequestedState;  

  310 uint32 CIM_ComputerSystem.RequestStateChange (  

  311     [IN] uint16 RequestedState,  

  312     [OUT] REF CIM_ConcreteJob Job,  

  313     [IN] datetime TimeoutPeriod );
```

314 6.1.2.3 Behavior Requirements

315 6.1.2.3.1 Preconditions

316 `$instance` represents the targeted instance of CIM_ComputerSystem.

```
317 $instance=<CIM_ComputerSystem single instance>;
```

318 6.1.2.3.2 Pseudo Code

```
319 &smResetRSC ( $instance.getObjectPath() );
320 &smEnd;
```

321 6.1.3 Set

322 This section describes how to implement the `set` verb when it is applied to an instance of
323 `CIM_ComputerSystem`. Implementations may support the use of the `set` verb with
324 `CIM_ComputerSystem`.

325 The `set` verb is used to modify descriptive properties of the `CIM_ComputerSystem` instance.

326 6.1.3.1 General Usage of Set for a Single Property

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

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

332 6.1.3.1.1 Command Form

```
333 set <CIM_ComputerSystem single instance> <propertyname>=<propertyvalue>
```

334 6.1.3.1.2 CIM Requirements

335 See `CIM_ComputerSystem` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
336 mandatory properties.

337 6.1.3.1.3 Behavior Requirements

```
338 $instance=<CIM_ComputerSystem single instance>
339 #propertyNames[] = {<propertyname>};
340 #propertyValues[] = {<propertyvalue>};
341 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
342 &smEnd;
```

343 6.1.3.2 General Usage of Set for Multiple Properties

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

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

350 6.1.3.2.1 Command Form

```
351 set <CIM_ComputerSystem single instance> <propertyname1>=<propertyvalue1>
352 <propertynamen>=<propertyvaluen>
```

353 6.1.3.2.2 CIM Requirements

354 See `CIM_ComputerSystem` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
355 mandatory properties.

356 6.1.3.2.3 Behavior Requirements

```

357 $instance=<CIM_ComputerSystem single instance>
358 #propertyNames[] = {<propertyname>};
359 for #i < n
360     {
361         #propertyNames[#i] = <propertyname#i>
362         #propertyValues[#i] = <propertyvalue#i>
363     }
364 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
365 &smEnd;
```

366 6.1.3.3 Set RequestedState to “Offline”

367 This section describes how to change the state of the Service Processor represented by
 368 CIM_ComputerSystem to “Enabled but Offline”.

369 6.1.3.3.1 Command Form

```

370 set <CIM_ComputerSystem single instance> RequestedState="Offline"
```

371 6.1.3.3.2 CIM Requirements

372 See CIM_ComputerSystem in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
 373 mandatory properties.

374 6.1.3.3.3 Behavior Requirements

375 6.1.3.3.3.1 Preconditions

376 \$instance represents the targeted instance of CIM_ComputerSystem.

```

377 $instance=<CIM_ComputerSystem single instance>;
```

378 6.1.3.3.3.2 Pseudo Code

```

379 //“Offline” is valuemap 6
380 &smRequestStateChange ( $instance.getObjectPath(), 6 );
381 &smEnd;
```

382 6.1.3.4 Set Service Processor Time

383 This section describes how to set the time on the Service Processor. There are two possible property
 384 name/value pairs that will result in the setting of the Service Processor time. When the `currenttime`
 385 property is specified, the format is expected to be in regular date-time format as defined in [DSP0216](#).
 386 When the `currenttime#time` property is specified, the format for the property value is defined in
 387 [DSP0216](#).

388 6.1.3.4.1 Command Form

```

389 set <CIM_ComputerSystem single instance>
390     ( currenttime=<mof time value> / currenttime#time=<friendly time value>)
```

391 6.1.3.4.2 CIM Requirements

392 See CIM_ComputerSystem and CIM_TimeService.ManageTime() in the “CIM Elements” section of the
 393 [Service Processor Profile](#) for the list of mandatory properties.

394 **6.1.3.4.3 Behavior Requirements**395 **6.1.3.4.3.1 Preconditions**

```

396 // #requestedtime contains the datetime which corresponds to the value of the
397 // currenttime or currenttime#time property value
398 // $instance=<CIM_ComputerSystem Single Instance>
399 // Find the TimeService associated to the target system
400 #Error = smOpAssociators(
401     $instance.getObjectPath(),
402     "CIM_ServiceAffectsElement",
403     "CIM_TimeService",
404     NULL,
405     NULL,
406     $Services[]);
407 //if there is not a service associated the function is not supported.
408 if (0 == $Services[].length){
409     //unsupported
410     $OperationError = smNewInstance("CIM_Error");
411     //CIM_ERR_NOT_SUPPORTED
412     $OperationError.CIMStatusCode = 7;
413     //Other
414     $OperationError.ErrorType = 1;
415     //Low
416     $OperationError.PerceivedSeverity = 2;
417     $OperationError.OwningEntity = DMTF:SMCLP;
418     $OperationError.MessageID = 0x00000001;
419     $OperationError.Message = "Operation is not supported.";
420     &smAddError($job, $OperationError);
421     &smMakeCommandStatus($job);
422     &smEnd;
423 }
424 $Service-> = $Services[0].getObjectPath();
425 %InArguments[] = { newArgument("GetRequest", FALSE)
426     newArgument("TimeData", #time),
427     newArgument("ManagedElement", $instance.getObjectPath())}
428 %OutArguments[] = { };
429 //invoke method
430 #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
431     "ManageTime",
432     %InArguments[],
433     %OutArguments[]);
434 // process return code to CLP Command Status
435 if (0 != #Error.code) {
436     //method invocation failed
437     if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
438         // if the method invocation contains an embedded error
439         // use it for the Error for the overall job
440         &smAddError($job, #Error.$error[0]);
441         &smMakeCommandStatus($job);

```

```

442     &smEnd;
443 }
444 else if (#Error.code == 17) {
445 //trap for CIM_METHOD_NOT_FOUND
446 //and make nice Unsupported msg.
447 //unsupported
448 $OperationError = smNewInstance("CIM_Error");
449 //CIM_ERR_NOT_SUPPORTED
450 $OperationError.CIMStatusCode = 7;
451 //Other
452 $OperationError.ErrorType = 1;
453 //Low
454 $OperationError.PerceivedSeverity = 2;
455 $OperationError.OwningEntity = DMTF:SMCLP;
456 $OperationError.MessageID = 0x00000001;
457 $OperationError.Message = "Operation is not supported.";
458 &smAddError($job, $OperationError);
459 &smMakeCommandStatus($job);
460 &smEnd;
461 }
462 else {
463 //operation failed, but no detailed error instance, need to make one up
464 //make an Error instance and associate with job for Operation
465 $OperationError = smNewInstance("CIM_Error");
466 //CIM_ERR_FAILED
467 $OperationError.CIMStatusCode = 1;
468 //Software Error
469 $OperationError.ErrorType = 4;
470 //Unknown
471 $OperationError.PerceivedSeverity = 0;
472 $OperationError.OwningEntity = DMTF:SMCLP;
473 $OperationError.MessageID = 0x00000009;
474 $OperationError.Message = "An internal software error has occurred.";
475 &smAddError($job, $OperationError);
476 &smMakeCommandStatus($job);
477 &smEnd;
478 }
479 }//if CIM op failed
480 else if (0 == #returnStatus) {
481 //completed successfully
482 %InArguments[] = { newArgument("GetRequest", FALSE)
483                   newArgument("ManagedElement", $instance.getObjectPath())}
484 %OutArguments[] = { newArgument("TimeData", #timedata)};
485 //invoke method
486 #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
487 "ManageTime",
488 %InArguments[],
489 %OutArguments[]);
490 if (0 != #Error.code) {

```



```

491 //method invocation failed
492 if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) ) {
493 // if the method invocation contains an embedded error
494 // use it for the Error for the overall job
495 &smAddError($job, #Error.$error[0]);
496 &smMakeCommandStatus($job);
497 &smEnd;
498 }
499 }
500 else {
501 //generic failure
502 $OperationError = smNewInstance("CIM_Error");
503 //CIM_ERR_FAILED
504 $OperationError.CIMStatusCode = 1;
505 //Other
506 $OperationError.ErrorType = 1;
507 //Low
508 $OperationError.PerceivedSeverity = 2;
509 $OperationError.OwningEntity = DMTF:SMCLP;
510 $OperationError.MessageID = 0x00000002;
511 $OperationError.Message = "Failed. No further information is available.";
512 &smAddError($job, $OperationError);
513 &smMakeCommandStatus($job);
514 }
515 &smEnd;
516 }
517 else if (1 == #returnStatus) {
518 //unsupported
519 $OperationError = smNewInstance("CIM_Error");
520 //CIM_ERR_NOT_SUPPORTED
521 $OperationError.CIMStatusCode = 7;
522 //Other
523 $OperationError.ErrorType = 1;
524 //Low
525 $OperationError.PerceivedSeverity = 2;
526 $OperationError.OwningEntity = DMTF:SMCLP;
527 $OperationError.MessageID = 0x00000001;
528 $OperationError.Message = "Operation is not supported.";
529 &smAddError($job, $OperationError);
530 &smMakeCommandStatus($job);
531 &smEnd;
532 }
533 else {
534 //generic failure
535 $OperationError = smNewInstance("CIM_Error");
536 //CIM_ERR_FAILED
537 $OperationError.CIMStatusCode = 1;
538 //Other
539 $OperationError.ErrorType = 1;

```

```

540 //Low
541 $OperationError.PerceivedSeverity = 2;
542 $OperationError.OwningEntity = DMTF:SMCLP;
543 $OperationError.MessageID = 0x00000002;
544 $OperationError.Message = "Failed. No further information is available.";
545 &smAddError($job, $OperationError);
546 &smMakeCommandStatus($job);
547 }

```

548 6.1.4 Show

549 This section describes how to implement the `show` verb when applied to an instance of
550 `CIM_ComputerSystem`. Implementations shall support the use of the `show` verb with
551 `CIM_ComputerSystem`.

552 6.1.4.1 Show Command Form for Multiple Instances Target

553 This command form is used to show many instances of `CIM_ComputerSystem`.

554 6.1.4.1.1 Command Form

```
555 show <CIM_ComputerSystem multiple instances>
```

556 6.1.4.1.2 CIM Requirements

557 See `CIM_ComputerSystem` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
558 mandatory properties.

559 6.1.4.1.3 Behavior Requirements

560 6.1.4.1.3.1 Preconditions

561 `$containerInstance` represents the instance of a subclass of `CIM_System` which represents the
562 container system and is associated to the targeted instances of `CIM_ComputerSystem` through the
563 `CIM_SystemComponent` association.

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

565 6.1.4.1.3.2 Pseudo Code

```

566 #propertylist[] = NULL;
567 if ( false == #all )
568 {
569     #propertylist[] = { <all mandatory non-key properties>
570 }
571 // Step 1 - find all the scoped instances
572 #Error = &smOpAssociators (
573     $containerInstance.getObjectPath(),
574     "SystemComponent",
575     "CIM_ComputerSystem",
576     NULL,
577     NULL,
578     NULL,
579     $instances[] );

```

```

580 if (0 != #Error.code)
581 {
582     &smProcessOpError (#Error);
583     //includes &smEnd;
584 }
585 // Step 2 - add their referenced properties
586 for $instance in $instances[] {
587     &lAddReferencedProperties (
588         $instance,
589         #referencedPropertyName[] );
590 }
591 //step 3 - display them
592 &smShowInstancesWithReferencedProperties (
593     $instances[],
594     #propertyList[],
595     #referencedPropertyName[] );
596 &smEnd;

```

597 6.1.4.2 Show Command Form for a Single Instance Target

598 This command form is used to show a single instance of CIM_ComputerSystem.

599 6.1.4.2.1 Command Form

```
600 show <CIM_ComputerSystem single instance>
```

601 6.1.4.2.2 CIM Requirements

602 See CIM_ComputerSystem in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
603 mandatory properties.

604 6.1.4.2.3 Behavior Requirements

605 6.1.4.2.3.1 Preconditions

606 \$instance represents the targeted instance of CIM_ComputerSystem.

```
607 $instance=<CIM_ComputerSystem single instance>;
```

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

609 6.1.4.2.3.2 Pseudo Code

```

610 #propertylist[] = NULL;
611 if ( false == #all ) {
612     #propertylist[] = <array of mandatory non-key property names (see CIM
613         Requirements)>;
614 }
615 &lAddReferencedProperties ( $instance, $referencedPropertyName[], #all );
616 &smShowInstanceWithReferencedProperties ( $device, #propertyList[],
617     $referencedPropertyName[] );
618 &smEnd;

```

619 **6.1.5 Start**

620 This section describes how to implement the `start` verb when applied to an instance of
 621 `CIM_ComputerSystem`. Implementations may support the use of the `start` verb with
 622 `CIM_ComputerSystem`.

623 **6.1.5.1 Command Form**

```
624 start <CIM_ComputerSystem single instance>
```

625 **6.1.5.2 CIM Requirements**

```
626 uint16 EnabledState;
627 uint16 RequestedState;
628 uint32 CIM_ComputerSystem.RequestStateChange (
629     [IN] uint16 RequestedState,
630     [OUT] REF CIM_ConcreteJob Job,
631     [IN] datetime TimeoutPeriod );
```

632 **6.1.5.3 Behavior Requirements**633 **6.1.5.3.1 Preconditions**

634 `$instance` represents the targeted instance of `CIM_ComputerSystem`.

```
635 $instance=<CIM_ComputerSystem single instance>;
```

636 **6.1.5.3.1.1 Pseudo Code**

```
637 &smStartRSC ( $instance.getObjectPath() );
638 &smEnd;
```

639 **6.1.6 Stop**

640 This section describes how to implement the `stop` verb when applied to an instance of
 641 `CIM_ComputerSystem`. Implementations may support the use of the `stop` verb with
 642 `CIM_ComputerSystem`.

643 **6.1.6.1 Command Form**

```
644 stop <CIM_ComputerSystem single instance>
```

645 **6.1.6.2 CIM Requirements**

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

652 **6.1.6.3 Behavior Requirements**653 **6.1.6.3.1 Preconditions**

654 `$instance` represents the targeted instance of `CIM_ComputerSystem`.

```
655 $instance=<CIM_ComputerSystem single instance>;
```

656 **6.1.6.3.1.2 Pseudo Code**

```
657 &smStopRSC ( $instance.getObjectPath() );
658 &smEnd;
```

659 **6.2 CIM_ElementCapabilities**

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

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

666 **Table 3 – Command Verb Requirements for CIM_ElementCapabilities**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.2.2.
Start	Not supported	
Stop	Not supported	

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

669 **6.2.1 Ordering of Results**

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

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

673 **6.2.2 Show**

674 This section describes how to implement the `show` verb when applied to an instance of
 675 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with
 676 `CIM_ElementCapabilities`.

677 **6.2.2.1 Show Command Form for Multiple Instances Target –**
 678 **CIM_EnabledLogicalElementCapabilities Reference**

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

682 **6.2.2.1.1 Command Form**

```
683 show <CIM_ElementCapabilities multiple instances>
```

684 6.2.2.1.2 CIM Requirements

685 See CIM_ElementCapabilities in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
686 mandatory properties.

687 6.2.2.1.3 Behavior Requirements

688 6.2.2.1.3.1 Preconditions

689 \$instance represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by
690 CIM_ElementCapabilities.

691 6.2.2.1.3.2 Pseudo Code

```
692 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
693 &smEnd;
```

694 6.2.2.2 Show Command Form for a Single Instance – CIM_ComputerSystem Reference

695 This command form is used to show a single instance of CIM_ElementCapabilities. This command form
696 corresponds to a show command issued against a single instance of CIM_ElementCapabilities where
697 only one reference is specified and the reference is to the instance of CIM_ComputerSystem.

698 6.2.2.2.1 Command Form

```
699 show <CIM_ElementCapabilities single instance>
```

700 6.2.2.2.2 CIM Requirements

701 See CIM_ElementCapabilities in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
702 mandatory properties.

703 6.2.2.2.3 Behavior Requirements

704 6.2.2.2.3.1 Preconditions

705 \$instance represents the instance of CIM_ComputerSystem which is referenced by
706 CIM_ElementCapabilities.

707 6.2.2.2.3.2 Pseudo Code

```
708 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
709 &smEnd;
```

710 6.2.2.3 Show Command Form for a Single Instance Target – Both References

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

714 6.2.2.3.1 Command Form

```
715 show <CIM_ElementCapabilities single instance>
```

716 6.2.2.3.2 CIM Requirements

717 See CIM_ElementCapabilities in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
718 mandatory properties.

719 6.2.2.3.3 Behavior Requirements

720 6.2.2.3.3.1 Preconditions

721 \$instanceA represents the referenced instance of CIM_ComputerSystem through the
 722 CIM_ElementCapabilities association. \$instanceB represents the instance of
 723 CIM_EnabledLogicalElementCapabilities which is referenced by CIM_ElementCapabilities.

724 6.2.2.3.3.2 Pseudo Code

```
725 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
726     $instanceB.getObjectPath() );
727 &smEnd;
```

728 6.3 CIM_EnabledLogicalElementCapabilities

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

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

735 **Table 4 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.3.2.
Start	Not supported	
Stop	Not supported	

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

738 6.3.1 Ordering of Results

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

- 741 • Results for CIM_EnabledLogicalElementCapabilities are unordered; therefore, no algorithm is
 742 defined.

743 6.3.2 Show

744 This section describes how to implement the `show` verb when applied to an instance of
745 `CIM_EnabledLogicalElementCapabilities`. Implementations shall support the use of the `show` verb with
746 `CIM_EnabledLogicalElementCapabilities`.

747 6.3.2.1 Show Command Form for Multiple Instances Target

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

749 6.3.2.1.1 Command Form

```
750 show <CIM_EnabledLogicalElementCapabilities multiple instances>
```

751 6.3.2.1.2 CIM Requirements

752 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Service Processor](#)
753 [Profile](#) for the list of mandatory properties.

754 6.3.2.1.3 Behavior Requirements

755 6.3.2.1.3.1 Preconditions

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

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

760 6.3.2.1.3.2 Pseudo Code

```
761 #propertylist[] = NULL;  
762 if ( false == #all )  
763 {  
764     #propertylist[] = <array of mandatory non-key property names (see CIM  
765     Requirements)>;  
766 }  
767 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",  
768     $containerInstance.getObjectPath(), #propertylist[] );  
769 &smEnd;
```

770 6.3.2.2 Show Command Form for a Single Instance Target

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

772 6.3.2.2.1 Command Form

```
773 show <CIM_EnabledLogicalElementCapabilities single instance>
```

774 6.3.2.2.2 CIM Requirements

775 See `CIM_EnabledLogicalElementCapabilities` in the “CIM Elements” section of the [Service Processor](#)
776 [Profile](#) for the list of mandatory properties.

777 **6.3.2.2.3 Behavior Requirements**

778 **6.3.2.2.3.1 Preconditions**

779 \$instance represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

```
780 $instance=<CIM_EnabledLogicalElementCapabilities single instance>;
```

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

782 **6.3.2.2.3.2 Pseudo Code**

```
783 #propertylist[] = NULL;
784 if ( false == #all )
785 {
786     #propertylist[] = <array of mandatory non-key property names (see CIM
787         Requirements)>;
788 }
789 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
790 &smEnd;
```

791 **6.4 CIM_HostedService**

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

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

798 **Table 5 – 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	
set	Not supported	
show	Shall	See 6.4.2.
start	Not supported	
stop	Not supported	

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

801 **6.4.1 Ordering of Results**

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

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

805 6.4.2 Show

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

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

809 6.4.2.1 Show Multiple Instances

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

813 6.4.2.1.1 Command Form

```
814 show <CIM_HostedService multiple instances>
```

815 6.4.2.1.2 CIM Requirements

816 See `CIM_HostedService` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
817 mandatory properties.

818 6.4.2.1.3 Behavior Requirements

819 6.4.2.1.3.1 Preconditions

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

821 6.4.2.1.3.2 Pseudo Code

```
822 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );  
823 &smEnd;
```

824 6.4.2.2 Show a Single Instance – CIM_TimeService Reference

825 This command form is for the `show` verb applied to a single instance. This command form corresponds to
826 a `show` command issued against `CIM_HostedService` where the reference specified is to an instance of
827 `CIM_TimeService`. An instance of `CIM_TimeService` is referenced by exactly one instance of
828 `CIM_HostedService`. Therefore, a single instance will be returned.

829 6.4.2.2.1 Command Form

```
830 show <CIM_HostedService single instance>
```

831 6.4.2.2.2 CIM Requirements

832 See `CIM_HostedService` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
833 mandatory properties.

834 6.4.2.2.3 Behavior Requirements

835 6.4.2.2.3.1 Preconditions

836 `$instance` contains the instance of `CIM_TimeService` which is referenced by `CIM_HostedService`.

837 6.4.2.2.3.2 Pseudo Code

```
838 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );  
839 &smEnd;
```

840 **6.4.2.3 Show a Single Instance – Both References**

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

844 **6.4.2.3.1 Command Form**

845 `show <CIM_HostedService single instance>`

846 **6.4.2.3.2 CIM Requirements**

847 See `CIM_HostedService` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
 848 mandatory properties.

849 **6.4.2.3.3 Behavior Requirements**

850 **6.4.2.3.3.1 Preconditions**

851 `$instanceA` contains the instance of `CIM_ComputerSystem` which is referenced by
 852 `CIM_HostedService`.

853 `$instanceB` contains the instance of `CIM_TimeService` which is referenced by `CIM_HostedService`.

854 **6.4.2.3.3.2 Pseudo Code**

```
855 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),
856     $instanceB.getObjectPath() );
857 &smEnd;
```

858 **6.5 CIM_IsSpare**

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

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

865 **Table 6 – Command Verb Requirements for CIM_IsSpare**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.5.2.
Start	Not supported	
Stop	Not supported	

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

868 6.5.1 Ordering of Results

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

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

872 6.5.2 Show

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

875 6.5.2.1 Show Command Form for Multiple Instances Target – CIM_RedundancySet Reference

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

879 6.5.2.1.1 Command Form

```
880 show <CIM_IsSpare multiple instances>
```

881 6.5.2.1.2 CIM Requirements

882 See CIM_IsSpare in the “CIM Elements” section of the [Service Processor Profile](#) for the list of mandatory
883 properties.

884 6.5.2.1.3 Behavior Requirements

885 6.5.2.1.3.1 Preconditions

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

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

888 6.5.2.1.3.2 Pseudo Code

```
889 #propertylist[] = NULL;
890 if ( false == #all )
891 {
892     #propertylist[] = <array of mandatory non-key property names (see CIM
893     Requirements)>;
894 }
895 &smShowAssociationInstances ( "CIM_IsSpare", $instance.getObjectPath(),
896     #propertylist[] );
897 &smEnd;
```

898 6.5.2.2 Show Command Form for a Single Instance – CIM_ComputerSystem Reference

899 This command form is used to show a single instance of CIM_IsSpare. This command form corresponds
900 to a `show` command issued against a single instance of CIM_IsSpare where only one reference is
901 specified and the reference is to the instance of CIM_ComputerSystem.

902 6.5.2.2.1 Command Form

```
903 show <CIM_IsSpare single instance>
```

904 **6.5.2.2.2 CIM Requirements**

905 See CIM_IsSpare in the “CIM Elements” section of the [Service Processor Profile](#) for the list of mandatory
906 properties.

907 **6.5.2.2.3 Behavior Requirements**

908 **6.5.2.2.3.1 Preconditions**

909 \$instance represents the instance of CIM_ComputerSystem which is referenced by CIM_IsSpare.

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

911 **6.5.2.2.3.2 Pseudo Code**

```
912 #propertylist[] = NULL;
913 if ( false == #all )
914 {
915     #propertylist[] = <array of mandatory non-key property names (see CIM
916     Requirements)>;
917 }
918 &smShowAssociationInstances ( "CIM_IsSpare", $instance.getObjectPath(),
919     #propertylist[]);
920 &smEnd;
```

921 **6.5.2.3 Show Command Form for a Single Instance Target – Both References**

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

925 **6.5.2.3.1 Command Form**

```
926 show <CIM_IsSpare single instance>
```

927 **6.5.2.3.2 CIM Requirements**

928 See CIM_IsSpare in the “CIM Elements” section of the [Service Processor Profile](#) for the list of mandatory
929 properties.

930 **6.5.2.3.3 Behavior Requirements**

931 **6.5.2.3.3.1 Preconditions**

932 \$instanceA represents the referenced instance of CIM_ComputerSystem through the CIM_IsSpare
933 association.

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

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

936 **6.5.2.3.3.2 Pseudo Code**

```

937 #propertylist[] = NULL;
938 if ( false == #all )
939     {
940         #propertylist[] = <array of mandatory non-key property names (see CIM
941             Requirements)>;
942     }
943 &smShowAssociationInstance ( "CIM_IsSpare", $instanceA.getObjectPath(),
944     $instanceB.getObjectPath(), #propertylist[] );
945 &smEnd;

```

946 **6.6 CIM_MemberOfCollection**

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

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

953 **Table 7 – Command Verb Requirements for CIM_MemberOfCollection**

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	Not supported	
Show	Shall	See 6.6.2.
Start	Not supported	
Stop	Not supported	

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

956 **6.6.1 Ordering of Results**

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

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

960 **6.6.2 Show**

961 This section describes how to implement the `show` verb when applied to an instance of
 962 `CIM_MemberOfCollection`. Implementations shall support the use of the `show` verb with
 963 `CIM_MemberOfCollection`.

964 **6.6.2.1 Show Command Form for Multiple Instances Target – CIM_RedundancySet Reference**

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

968 **6.6.2.1.1 Command Form**

```
969 show <CIM_MemberOfCollection multiple instances>
```

970 **6.6.2.1.2 CIM Requirements**

971 See CIM_MemberOfCollection in the “CIM Elements” section of the [Service Processor Profile](#) for the list
972 of mandatory properties.

973 **6.6.2.1.3 Behavior Requirements**

974 **6.6.2.1.3.1 Preconditions**

975 `$instance` represents the instance of CIM_RedundancySet which is referenced by
976 CIM_MemberOfCollection.

977 **6.6.2.1.3.2 Pseudo Code**

```
978 &smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.getObjectPath() );  
979 &smEnd;
```

980 **6.6.2.2 Show Command Form for a Single Instance – CIM_ComputerSystem Reference**

981 This command form is used to show a single instance of CIM_MemberOfCollection. This command form
982 corresponds to a `show` command issued against a single instance of CIM_MemberOfCollection where
983 only one reference is specified and the reference is to the instance of CIM_ComputerSystem.

984 **6.6.2.2.1 Command Form**

```
985 show <CIM_MemberOfCollection single instance>
```

986 **6.6.2.2.2 CIM Requirements**

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

989 **6.6.2.2.3 Behavior Requirements**

990 **6.6.2.2.3.1 Preconditions**

991 `$instance` represents the instance of CIM_ComputerSystem which is referenced by
992 CIM_MemberOfCollection.

993 **6.6.2.2.3.2 Pseudo Code**

```
994 &smShowAssociationInstances ( "CIM_MemberOfCollection", $instance.getObjectPath() );  
995 &smEnd;
```

996 **6.6.2.3 Show Command Form for a Single Instance Target – Both References**

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

1000 **6.6.2.3.1 Command Form**1001 `show <CIM_MemberOfCollection single instance>`1002 **6.6.2.3.2 CIM Requirements**1003 See CIM_MemberOfCollection in the “CIM Elements” section of the [Service Processor Profile](#) for the list
1004 of mandatory properties.1005 **6.6.2.3.3 Behavior Requirements**1006 **6.6.2.3.3.1 Preconditions**1007 \$instanceA represents the referenced instance of CIM_ComputerSystem through
1008 CIM_MemberOfCollection association.1009 \$instanceB represents the instance of CIM_RedundancySet which is referenced by
1010 CIM_MemberOfCollection.1011 **6.6.2.3.3.2 Pseudo Code**1012 `&smShowAssociationInstance ("CIM_MemberOfCollection", $instanceA.getObjectPath(),`
1013 `$instanceB.getObjectPath());`
1014 `&smEnd;`1015 **6.7 CIM_OwningCollectionElement**1016 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).1017 Table 8 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1018 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1019 target. Table 8 is for informational purposes only; in case of a conflict between Table 8 and requirements
1020 detailed in the following sections, the text detailed in the following sections supersedes the information in
1021 Table 8.1022 **Table 8 – Command Verb Requirements for CIM_OwningCollectionElement**

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

1023 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1024 `reset`, `set`, `start`, and `stop`.

1025 6.7.1 Ordering of Results

1026 When results are returned for multiple instances of CIM_OwningCollectionElement, implementations shall
1027 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1028 • Results for CIM_OwningCollectionElement are unordered; therefore, no algorithm is defined.

1029 6.7.2 Show

1030 This section describes how to implement the `show` verb when applied to an instance of
1031 CIM_OwningCollectionElement. Implementations shall support the use of the `show` verb with
1032 CIM_OwningCollectionElement.

1033 6.7.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

1034 This command form is used to show many instances of CIM_OwningCollectionElement. This command
1035 form corresponds to a `show` command issued against the instance of CIM_OwningCollectionElement
1036 where only one reference is specified and the reference is to the scoping instance of
1037 CIM_ComputerSystem.

1038 6.7.2.1.1 Command Form

```
1039 show <CIM_OwningCollectionElement multiple instances>
```

1040 6.7.2.1.2 CIM Requirements

1041 See CIM_OwningCollectionElement in the “CIM Elements” section of the [Service Processor Profile](#) for the
1042 list of mandatory properties.

1043 6.7.2.1.3 Behavior Requirements

1044 6.7.2.1.3.1 Preconditions

1045 `$instance` represents the instance of a CIM_ComputerSystem, which is referenced by
1046 CIM_OwningCollectionElement.

1047 6.7.2.1.3.2 Pseudo Code

```
1048 &smShowAssociationInstances ( "CIM_OwningCollectionElement",  
1049     $instance.getObjectPath() );  
1050 &smEnd;
```

1051 6.7.2.2 Show Command Form for a Single Instance Target – CIM_RedundancySet Reference

1052 This command form is used to show a single instance of CIM_OwningCollectionElement. This command
1053 form corresponds to a `show` command issued against a single instance of
1054 CIM_OwningCollectionElement, where only one reference is specified and the reference is to an instance
1055 of CIM_RedundancySet.

1056 6.7.2.2.1 Command Form

```
1057 show <CIM_OwningCollectionElement single instance>
```

1058 6.7.2.2.2 CIM Requirements

1059 See CIM_OwningCollectionElement in the “CIM Elements” section of the [Service Processor Profile](#) for the
1060 list of mandatory properties.

1061 **6.7.2.2.3 Behavior Requirements**1062 **6.7.2.2.3.1 Preconditions**

1063 \$instance represents the instance of CIM_RedundancySet which is referenced by
1064 CIM_OwningCollectionElement.

1065 **6.7.2.2.3.2 Pseudo Code**

```
1066 &smShowAssociationInstances ( "CIM_OwningCollectionElement",
1067     $instance.getObjectPath() );
1068 &smEnd;
```

1069 **6.7.2.3 Show Command Form for a Single Instance Target – Both References**

1070 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1071 a `show` command issued against CIM_OwningCollectionElement where both references are specified and
1072 therefore the desired instance is unambiguously identified.

1073 **6.7.2.3.1 Command Form**

```
1074 show <CIM_OwningCollectionElement single instance>
```

1075 **6.7.2.3.2 CIM Requirements**

1076 See CIM_OwningCollectionElement in the “CIM Elements” section of the [Service Processor Profile](#) for the
1077 list of mandatory properties.

1078 **6.7.2.3.3 Behavior Requirements**1079 **6.7.2.3.3.1 Preconditions**

1080 \$instanceA represents the referenced instance of CIM_RedundancySet through
1081 CIM_OwningCollectionElement association.

1082 \$instanceB represents the instance of CIM_ComputerSystem which is referenced by
1083 CIM_OwningCollectionElement.

1084 **6.7.2.3.3.2 Pseudo Code**

```
1085 &smShowAssociationInstance ( "CIM_OwningCollectionElement",
1086     $instanceA.getObjectPath(), $instanceB.getObjectPath() );
1087 &smEnd;
```

1088 **6.8 CIM_RedundancySet**

1089 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in [DSP0216](#).

1090 Table 9 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1091 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1092 target. Table 9 is for informational purposes only; in case of a conflict between Table 9 and requirements
1093 detailed in the following sections, the text detailed in the following sections supersedes the information in
1094 Table 9.

1095

Table 9 – Command Verb Requirements for CIM_RedundancySet

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	May	See 6.8.2.
Show	Shall	See 6.8.3.
Start	Not supported	
Stop	Not supported	

1096 No mapping is defined for the following verbs for the specified target: create, delete, dump, and load.

1097 **6.8.1 Ordering of Results**

1098 When results are returned for multiple instances of CIM_RedundancySet, implementations shall utilize
 1099 the following algorithm to produce the natural (that is, default) ordering:

- 1100 • Results for CIM_RedundancySet are unordered; therefore, no algorithm is defined.

1101 **6.8.2 Set**

1102 This section describes how to implement the `set` verb when it is applied to an instance of
 1103 CIM_RedundancySet. Implementations may support the use of the `set` verb with CIM_RedundancySet.

1104 The `set` verb is used to modify descriptive properties of the CIM_RedundancySet instance.

1105 **6.8.2.1 Set Command for Failovers**

1106 This section describes how to use the `set` verb when it is applied to an instance of CIM_RedundancySet
 1107 to failover from an active Service Processor to a spare Service Processor.

1108 **6.8.2.1.1 Command Form**

```
1109 set <CIM_RedundancySet single instance> failoverfrom=<CIM_ComputerSystem single
1110 instance> failovertto=<CIM_ComputerSystem single instance>
```

1111 **6.8.2.1.2 CIM Requirements**

```
1112 uint32 CIM_RedundancySet.Failover (
1113     [IN] REF CIM_ManagedElement FailoverFrom,
1114     [IN] REF CIM_ManagedElement FailoverTo );
```

1115 **6.8.2.1.3 Behavior Requirements**

1116 **6.8.2.1.3.1 Preconditions**

```
1117 $instance=<CIM_RedundancySet single instance>
1118 $FailoverFrom=<failoverfrom requested instance of CIM_ComputerSystem>
1119 $FailoverTo=<failovertto requested instance of CIM_ComputerSystem>
```

1120 6.8.2.1.3.2 Pseudo Code

```

1121 %InArguments[] = { newArgument ( "FailoverFrom", $FailoverFrom.getObjectPath() ),
1122                   newArgument ( "FailoverTo", $FailoverTo.getObjectPath () );
1123 %OutArguments[] = {};
1124 #Error = InvokeMethod ($target->,
1125                       "Failover",
1126                       %InArguments[],
1127                       %OutArguments[],
1128                       #returnStatus);
1129 if ( 0 != #Error.code)
1130 {
1131     //method invocation failed
1132     if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )
1133     {
1134         //if the method invocation contains an embedded error
1135         //use it for the Error for the overall job
1136         &smAddError($job, #Error.$error[0]);
1137         &smMakeCommandStatus($job);
1138         &smEnd;
1139     }
1140     else if ( 17 == #Error.code ) {
1141         //17 - CIM_ERR_METHOD_NOT_FOUND
1142         // The specified extrinsic method does not exist.
1143         $OperationError = smNewInstance("CIM_Error");
1144         // CIM_ERR_METHOD_NOT_FOUND
1145         $OperationError.CIMStatusCode = 17;
1146         //Software Error
1147         $OperationError.ErrorType = 10;
1148         //Unknown
1149         $OperationError.PerceivedSeverity = 0;
1150         $OperationError.OwningEntity = DMTF:SMCLP;
1151         $OperationError.MessageID = 0x00000001;
1152         $OperationError.Message = "Operation is not supported."
1153         &smAddError($job, $OperationError);
1154         &smMakeCommandStatus($job);
1155         &smEnd;
1156     }
1157     else
1158     {
1159         //operation failed, but no detailed error instance, need to make one up
1160         //make an Error instance and associate with job for Operation
1161         $OperationError = smNewInstance("CIM_Error");
1162         //CIM_ERR_FAILED
1163         $OperationError.CIMStatusCode = 1;
1164         //Software Error
1165         $OperationError.ErrorType = 4;
1166         //Unknown
1167         $OperationError.PerceivedSeverity = 0;
1168         $OperationError.OwningEntity = DMTF:SMCLP;

```

```
1169     $OperationError.MessageID = 0x00000009;
1170     $OperationError.Message = "An internal software error has occurred.";
1171     &smAddError($job, $OperationError);
1172     &smMakeCommandStatus($job);
1173     &smEnd;
1174 }
1175 }//if CIM op failed
1176 else if (0 == #returnStatus) {
1177     //completed successfully
1178     &smCommandCompleted($job);
1179     &smEnd;
1180 }
1181 else if (1 == #returnStatus) {
1182     //unsupported
1183     $OperationError = smNewInstance("CIM_Error");
1184     //CIM_ERR_NOT_SUPPORTED
1185     $OperationError.CIMStatusCode = 7;
1186     //Other
1187     $OperationError.ErrorType = 1;
1188     //Low
1189     $OperationError.PerceivedSeverity = 2;
1190     $OperationError.OwningEntity = DMTF:SMCLP;
1191     $OperationError.MessageID = 0x00000001;
1192     $OperationError.Message = "Operation is not supported.";
1193     &smAddError($job, $OperationError);
1194     &smMakeCommandStatus($job);
1195     &smEnd;
1196 }
1197 else if (2 == #returnStatus) {
1198     //generic failure
1199     $OperationError = smNewInstance("CIM_Error");
1200     //CIM_ERR_FAILED
1201     $OperationError.CIMStatusCode = 1;
1202     //Other
1203     $OperationError.ErrorType = 1;
1204     //Low
1205     $OperationError.PerceivedSeverity = 2;
1206     $OperationError.OwningEntity = DMTF:SMCLP;
1207     $OperationError.MessageID = 0x00000002;
1208     $OperationError.Message = "Failed. No further information is available.";
1209     &smAddError($job, $OperationError);
1210     &smMakeCommandStatus($job);
1211 }
1212 else {
1213     //unspecified return code, generic failure
1214     $OperationError = smNewInstance("CIM_Error");
1215     //CIM_ERR_FAILED
1216     $OperationError.CIMStatusCode = 1;
1217     //Other
```

```

1218     $OperationError.ErrorType = 1;
1219     //Low
1220     $OperationError.PerceivedSeverity = 2;
1221     $OperationError.OwningEntity = DMTF:SMCLP;
1222     $OperationError.MessageID = 0x00000002;
1223     $OperationError.Message = "Failed. No further information is available.";
1224     &smAddError($job, $OperationError);
1225     &smMakeCommandStatus($job);
1226     &smEnd;
1227 }

```

1228 6.8.2.2 General Usage of Set for a Single Property

1229 This command form corresponds to the general usage of the `set` verb to modify a single property of a
 1230 target instance. This is the most common case.

1231 The requirement for supporting modification of a property using this command form shall be equivalent to
 1232 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
 1233 in the [Service Processor Profile](#).

1234 6.8.2.2.1 Command Form

```

1235 set <CIM_RedundancySet single instance> <propertyname>=<propertyvalue>

```

1236 6.8.2.2.2 CIM Requirements

1237 See `CIM_RedundancySet` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
 1238 mandatory properties.

1239 6.8.2.2.3 Behavior Requirements

```

1240 $instance=<CIM_RedundancySet single instance>
1241 #propertyName[] = {<propertyname>};
1242 #propertyValues[] = {<propertyvalue>};
1243 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
1244 &smEnd;

```

1245 6.8.2.3 General Usage of Set for Multiple Properties

1246 This command form corresponds to the general usage of the `set` verb to modify multiple properties of a
 1247 target instance where there is not an explicit relationship between the properties. This is the most
 1248 common case.

1249 The requirement for supporting modification of a property using this command form shall be equivalent to
 1250 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
 1251 in the [Service Processor Profile](#).

1252 6.8.2.3.1 Command Form

```

1253 set <CIM_RedundancySet single instance> <propertyname1>=<propertyvalue1>
1254 <propertyname2>=<propertyvalue2>

```

1255 6.8.2.3.2 CIM Requirements

1256 See `CIM_RedundancySet` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
 1257 mandatory properties.

1258 6.8.2.3.3 Behavior Requirements

```

1259 $instance=<CIM_RedundancySet single instance>
1260 #propertyName[] = {<propertyname>};
1261 for #i < n
1262     {
1263         #propertyName[#i] = <propertyname#i>
1264         #propertyValue[#i] = <propertyvalue#i>
1265     }
1266 &smSetInstance ( $instance, #propertyName[], #propertyValue[] );
1267 &smEnd;

```

1268 6.8.3 Show

1269 This section describes how to implement the `show` verb when applied to an instance of
 1270 `CIM_RedundancySet`. Implementations shall support the use of the `show` verb with `CIM_RedundancySet`.

1271 6.8.3.1 Show Command Form for Multiple Instances Target

1272 This command form is used to show many instances of `CIM_RedundancySet`.

1273 6.8.3.1.1 Command Form

```

1274 show <CIM_RedundancySet multiple instances>

```

1275 6.8.3.1.2 CIM Requirements

1276 See `CIM_RedundancySet` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
 1277 mandatory properties.

1278 6.8.3.1.3 Behavior Requirements

1279 6.8.3.1.3.1 Preconditions

1280 `$containerInstance` represents the instance of `CIM_ComputerSystem` which represents the
 1281 container system and is associated to the targeted instances of `CIM_RedundancySet` through the
 1282 `CIM_OwningCollectionElement` association.

1283 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1284 6.8.3.1.3.2 Pseudo Code

```

1285 #propertylist[] = NULL;
1286 if ( false == #all )
1287     {
1288         #propertylist[] = <array of mandatory non-key property names (see CIM
1289             Requirements)>;
1290     }
1291 &smShowInstances ( "CIM_RedundancySet", "CIM_OwningCollectionElement",
1292     $containerInstance.getObjectPath(), #propertylist[] );
1293 &smEnd;

```

1294 6.8.3.2 Show Command Form for a Single Instance Target

1295 This command form is used to show a single instance of `CIM_RedundancySet`.

1296 **6.8.3.2.1 Command Form**1297 `show <CIM_RedundancySet single instance>`1298 **6.8.3.2.2 CIM Requirements**1299 See CIM_RedundancySet in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
1300 mandatory properties.1301 **6.8.3.2.3 Behavior Requirements**1302 **6.8.3.2.3.1 Preconditions**1303 In this section `$instance` represents the targeted instance of CIM_RedundancySet.1304 `$instance=<CIM_RedundancySet single instance>;`1305 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.1306 **6.8.3.2.3.2 Pseudo Code**

```

1307 #propertylist[] = NULL;
1308 if ( false == #all )
1309     {
1310         #propertylist[] = <array of mandatory non-key property names (see CIM
1311             Requirements)>;
1312     }
1313 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
1314 &smEnd;

```

1315 **6.9 CIM_ServiceAffectsElement**1316 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1317 Table 10 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1318 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1319 target. Table 10 is for informational purposes only; in case of a conflict between Table 10 and
 1320 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1321 information in Table 10.

1322 **Table 10 – 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 6.9.2.
start	Not supported	
stop	Not supported	

1323 No mappings are defined for the following verbs for the specified target: create, delete, dump, load,
 1324 reset, set, start, and stop.

1325 6.9.1 Ordering of Results

1326 When results are returned for multiple instances of CIM_ServiceAffectsElement, implementations shall
1327 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1328 • Results for CIM_ServiceAffectsElement are unordered; therefore, no algorithm is defined.

1329 6.9.2 Show

1330 This section describes how to implement the `show` verb when applied to an instance of
1331 CIM_ServiceAffectsElement. Implementations shall support the use of the `show` verb with
1332 CIM_ServiceAffectsElement.

1333 The `show` command is used to display information about the CIM_ServiceAffectsElement instance or
1334 instances.

1335 6.9.2.1 Show a Single Instance – CIM_TimeService Reference

1336 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1337 a `show` command issued against CIM_ServiceAffectsElement where only one reference is specified and
1338 the reference is to an instance of CIM_TimeService.

1339 6.9.2.1.1 Command Form

```
1340 show <CIM_ServiceAffectsElement multiple objects>
```

1341 6.9.2.1.2 CIM Requirements

1342 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Service Processor Profile](#) for the list
1343 of mandatory properties.

1344 6.9.2.1.3 Behavior Requirements

1345 6.9.2.1.3.1 Preconditions

1346 `$instance` contains the instance of CIM_TimeService which is referenced by
1347 CIM_ServiceAffectsElement.

1348 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1349 6.9.2.1.3.2 Pseudo Code

```
1350 #propertylist[] = NULL;
1351 if ( false == #all )
1352 {
1353     #propertylist[] = {/all mandatory non-key properties};
1354 }
1355 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),
1356     #propertylist[] );
1357 &smEnd;
```

1358 6.9.2.2 Show Multiple Instances – CIM_ComputerSystem Reference

1359 This command form is for the `show` verb applied to a single instance. This command form corresponds to
1360 a `show` command issued against CIM_ServiceAffectsElement where the reference specified is to an
1361 instance of CIM_ComputerSystem.

1362 **6.9.2.2.1 Command Form**1363

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

1364 **6.9.2.2.2 CIM Requirements**1365 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Service Processor Profile](#) for the list
1366 of mandatory properties.1367 **6.9.2.2.3 Behavior Requirements**1368 **6.9.2.2.3.1 Preconditions**1369 \$instance contains the instance of CIM_ComputerSystem which is referenced by
1370 CIM_ServiceAffectsElement.

1371 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1372 **6.9.2.2.3.2 Pseudo Code**1373

```
#propertylist[] = NULL;  
1374 if ( false == #all )  
1375 {  
1376     #propertylist[] = {/all mandatory non-key properties};  
1377 }  
1378 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),  
1379     #propertylist[] );  
1380 &smEnd;
```

1381 **6.9.2.3 Show a Single Instance – Both References**1382 This command form is for the show verb applied to a single instance. This command form corresponds to
1383 a show command issued against CIM_ServiceAffectsElement where both references are specified and
1384 therefore the desired instance is unambiguously identified.1385 **6.9.2.3.1 Command Form**1386

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

1387 **6.9.2.3.2 CIM Requirements**1388 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Service Processor Profile](#) for the list
1389 of mandatory properties.1390 **6.9.2.3.3 Behavior Requirements**1391 **6.9.2.3.3.1 Preconditions**1392 \$instanceA contains the instance of CIM_ServiceAvailableToElement which is referenced by
1393 CIM_ServiceAffectsElement.1394 \$instanceB contains the instance of CIM_ComputerSystem which is referenced by
1395 CIM_ServiceAffectsElement.

1396 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1397 **6.9.2.3.3.2 Pseudo Code**

```

1398 #propertylist[] = NULL;
1399 if ( false == #all )
1400     {
1401         #propertylist[] = {/all mandatory non-key properties};
1402     }
1403 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),
1404     $instanceB.getObjectPath(), #propertylist[] );
1405 &smEnd;
    
```

1406 **6.10 CIM_TimeService**

1407 The `cd` and `help` verbs shall be supported as described in [DSP0216](#).

1408 Table 11 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1409 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1410 target. Table 11 is for informational purposes only; in case of a conflict between Table 11 and
 1411 requirements detailed in the following sections, the text detailed in the following sections supersedes the
 1412 information in Table 11.

1413 **Table 11 – Command Verb Requirements for CIM_TimeService**

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 6.10.2.
start	Not supported	
stop	Not supported	

1414 No mappings are defined for the following verbs for the specified target: `create`, `delete`, `dump`, and
 1415 `load`.

1416 **6.10.1 Ordering of Results**

1417 When results are returned for multiple instances of `CIM_TimeService`, implementations shall utilize the
 1418 following algorithm to produce the natural (that is, default) ordering:

- 1419 • Results for `CIM_TimeService` are unordered; therefore, no algorithm is defined.

1420 **6.10.2 Show**

1421 This section describes how to implement the `show` verb when applied to an instance of
 1422 `CIM_TimeService`. Implementations shall support the use of the `show` verb with `CIM_TimeService`.

1423 The `show` verb is used to display information about the `CIM_TimeService` instance.

1424 6.10.2.1 Show a Single Instance

1425 This command form is for the `show` verb applied to a single instance of `CIM_TimeService`.

1426 6.10.2.1.1 Command Form

```
1427 show <CIM_TimeService single instance>
```

1428 6.10.2.1.2 CIM Requirements

1429 See `CIM_TimeService` in the “CIM Elements” section of the [Service Processor Profile](#) for the list of
1430 mandatory properties.

1431 6.10.2.1.3 Behavior Requirements**1432 6.10.2.1.3.1 Preconditions**

1433 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1434 6.10.2.1.3.2 Pseudo Code

```
1435 #propertylist[] = NULL;  
1436 if ( false == #all )  
1437     {  
1438         #propertylist[] = { //all mandatory non-key properties};  
1439     }  
1440 $instance=<CIM_TimeService single instance>  
1441 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );  
1442 &smEnd;
```

**ANNEX A
(informative)**

Change Log

1443
1444
1445
1446
1447

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

1448