



1
2
3
4

Document Number: DSP0823

Date: 2009-07-14

Version: 1.0.0

5 **Power State Management Profile to SM CLP**
6 **Mapping Specification**

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

10

11 Copyright notice

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

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

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

33

34

CONTENTS

35 Foreword 5

36 Introduction 6

37 1 Scope 7

38 2 Normative References..... 7

39 2.1 Approved References 7

40 2.2 Other References..... 7

41 3 Terms and Definitions..... 7

42 4 Symbols and Abbreviated Terms..... 8

43 5 Recipes..... 9

44 5.1 smRequestPowerStateChange 9

45 6 Mappings..... 12

46 6.1 CIM_ComputerSystem..... 12

47 6.2 CIM_PowerManagementService 25

48 6.3 CIM_PowerManagementCapabilities 27

49 6.4 CIM_AssociatedPowerManagementService 28

50 6.5 CIM_ElementCapabilities 31

51 6.6 CIM_HostedService 34

52 ANNEX A (informative) Change Log 38

53

54 Tables

55 Table 1 – Command Verb Requirements for CIM_ComputerSystem 12

56 Table 2 – Command Verb Requirements for CIM_PowerManagementService 25

57 Table 3 – Command Verb Requirements for CIM_PowerManagementCapabilities 27

58 Table 4 – Command Verb Requirements for CIM_AssociatedPowerManagementService 29

59 Table 5 – Command Verb Requirements for CIM_ElementCapabilities 32

60 Table 6 – Command Verb Requirements for CIM_HostedService 35

61

63

Foreword

64 The *Power State Management Profile to SM CLP Mapping Specification* (DSP0823) was prepared by the
65 Server Management Working Group.

66 **Conventions**

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

69 **Acknowledgements**

70 The authors wish to acknowledge the following participants from the DTMF Server Management Working
71 Group:

- 72 • RadhaKrishna R. Dasari – Dell
- 73 • Khachatur Papanyan – Dell
- 74 • Aaron Merkin – IBM
- 75 • Perry Vincent – Intel

76

77

Introduction

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

82 The target audience for this specification is implementers of the SM CLP support for the [Power State](#)
83 [Management Profile](#).

84 Power State Management Profile to SM CLP Mapping 85 Specification

86 1 Scope

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

89 2 Normative References

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

93 2.1 Approved References

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

96 DMTF DSP1027, *Power State Management Profile 1.0*,
97 http://www.dmtf.org/standards/published_documents/DSP1027_1.0.pdf

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

100 2.2 Other References

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

103 3 Terms and Definitions

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

105 3.1

106 **can**

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

108 3.2

109 **cannot**

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

111 3.3

112 **conditional**

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

- 115 **3.4**
116 **mandatory**
117 indicates requirements to be followed strictly in order to conform to the document and from which no
118 deviation is permitted
- 119 **3.5**
120 **may**
121 indicates a course of action permissible within the limits of the document
- 122 **3.6**
123 **need not**
124 indicates a course of action permissible within the limits of the document
- 125 **3.7**
126 **optional**
127 indicates a course of action permissible within the limits of the document
- 128 **3.8**
129 **shall**
130 indicates requirements to be followed strictly in order to conform to the document and from which no
131 deviation is permitted
- 132 **3.9**
133 **shall not**
134 indicates requirements to be followed strictly in order to conform to the document and from which no
135 deviation is permitted
- 136 **3.10**
137 **should**
138 indicates that among several possibilities, one is recommended as particularly suitable, without
139 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required
- 140 **3.11**
141 **should not**
142 indicates that a certain possibility or course of action is deprecated but not prohibited
143

144 **4 Symbols and Abbreviated Terms**

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

- 146 **4.1**
147 **CIM**
148 Common Information Model
- 149 **4.2**
150 **CLP**
151 Command Line Protocol
- 152 **4.3**
153 **DMTF**
154 Distributed Management Task Force

155 **4.4**
 156 **SM**
 157 Server Management

158 **4.5**
 159 **SMI-S**
 160 Storage Management Initiative Specification

161 **4.6**
 162 **SNIA**
 163 Storage Networking Industry Association

164 **4.7**
 165 **UFsT**
 166 User Friendly Selection Tag
 167

168 **5 Recipes**

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

- 171 • smResetRSC
- 172 • smShowInstance
- 173 • smShowInstances
- 174 • smShowAssociationInstance
- 175 • smShowAssociationInstances
- 176 • smStartRSC
- 177 • smStopRSC

178 The following section details the Local Recipe defined for use in this mapping.

179 **5.1 smRequestPowerStateChange**

180 **5.1.1 Description**

181 This method is used to change the power state of a computer system.

182 **5.1.2 Pseudo Code**

```

183 sub void smRequestPowerStateChange (uint64 #PowerState, $targetSystem->, #time,
184     $targetPMS->)
185 {
186 //PowerState parameter contains the requested power state.
187 //$targetsystem-> parameter contains the target system whose power state
188 // needs to be set.
189 // #time parameter contains the time at which the request needs to be
190 // performed.
191 // targetPMS-> parameter contains the PowerManagementService responsible to
192 // perform the request.

```

```
193 $instanceConcreteJob = smNewInstance ("CIM_ConcreteJob");
194 %InArguments[] = {newArgument("PowerState", #PowerState),
195 newArgument("ManagedElement", $targetSystem->),
196 newArgument("Time", #time)}
197 %OutArguments[] = {newArgument("Job", $instanceConcreteJob.GetObjectPath())}
198 #Error = InvokeMethod ($targetPMS->,
199 "RequestPowerStateChange",
200 %InArguments[],
201 %OutArguments[],
202 #returnStatus);
203 if (0 != #Error.code)
204 {
205 //method invocation failed
206 if ( (null != #Error.$error) && (null != #Error.$error[0]) )
207 {
208 //if the method invocation contains an embedded error
209 //use it for the Error for the overall job
210 &smAddError($job, #Error.$error[0]);
211 &smMakeCommandStatus($job);
212 &smEnd;
213 }
214 else if (17 == #returnStatus) {
215 //The specified extrinsic method does not exist
216 $OperationError = smNewInstance("CIM_Error");
217 //CIM_ERR_METHOD_NOT_FOUND
218 $OperationError.CIMStatusCode = 17;
219 //Software Error
220 $OperationError.ErrorType = 10;
221 //Low
222 $OperationError.PerceivedSeverity = 0;
223 $OperationError.OwningEntity = DMTF:SMCLP;
224 $OperationError.MessageID = 0x00000001;
225 $OperationError.Message = "Operation is not supported";
226 &smAddError($job, $OperationError);
227 &smMakeCommandStatus($job);
228 }
229 else
230 {
231 //operation failed, but no detailed error instance, need to make //one up
232 //make an Error instance and associate with job for Operation
233 $OperationError = smNewInstance("CIM_Error");
234 //CIM_ERR_FAILED
235 $OperationError.CIMStatusCode = 1;
236 //Software Error
237 $OperationError.ErrorType = 4;
238 //Unknown
239 $OperationError.PerceivedSeverity = 0;
240 $OperationError.OwningEntity = DMTF:SMCLP;
241 $OperationError.MessageID = 0x00000009;
```

```
242     $OperationError.Message = "An internal software error has occurred.";
243     &smAddError($job, $OperationError);
244     &smMakeCommandStatus($job);
245     &smEnd;
246 }
247 }//if CIM op failed
248 else if (0 == #returnStatus) {
249     //completed successfully
250     &smCommandCompleted($job);
251     &smEnd;
252 }
253 else if (1 == #returnStatus) {
254     //unsupported
255     $OperationError = smNewInstance("CIM_Error");
256     //CIM_ERR_NOT_SUPPORTED
257     $OperationError.CIMStatusCode = 7;
258     //Other
259     $OperationError.ErrorType = 1;
260     //Low
261     $OperationError.PerceivedSeverity = 2;
262     $OperationError.OwningEntity = DMTF:SMCLP;
263     $OperationError.MessageID = 0x00000001;
264     $OperationError.Message = "Operation is not supported.";
265     &smAddError($job, $OperationError);
266     &smMakeCommandStatus($job);
267     &smEnd;
268 }
269 else if (2 == #returnStatus) {
270     //generic failure
271     $OperationError = smNewInstance("CIM_Error");
272     //CIM_ERR_FAILED
273     $OperationError.CIMStatusCode = 1;
274     //Other
275     $OperationError.ErrorType = 1;
276     //Low
277     $OperationError.PerceivedSeverity = 2;
278     $OperationError.OwningEntity = DMTF:SMCLP;
279     $OperationError.MessageID = 0x00000002;
280     $OperationError.Message = "Failed. No further information is available.";
281     &smAddError($job, $OperationError);
282     &smMakeCommandStatus($job);
283 }
284 else {
285     //unspecified return code, generic failure
286     $OperationError = smNewInstance("CIM_Error");
287     //CIM_ERR_FAILED
288     $OperationError.CIMStatusCode = 1;
289     //Other
290     $OperationError.ErrorType = 1;
```

```

291     //Low
292     $OperationError.PerceivedSeverity = 2;
293     $OperationError.OwningEntity = DMTF:SMCLP;
294     $OperationError.MessageID = 0x00000002;
295     $OperationError.Message = "Failed. No further information is available.";
296     &smAddError($job, $OperationError);
297     &smMakeCommandStatus($job);
298     &smEnd;
299 }

```

300 6 Mappings

301 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
 302 the [Power State Management Profile](#). Requirements specified here related to support for a CLP verb for a
 303 particular class are solely within the context of this profile.

304 6.1 CIM_ComputerSystem

305 CIM_ComputerSystem is not owned by the [Power State Management Profile](#). The following mappings are
 306 in addition to those stated by the profile which owns the CIM_ComputerSystem definition.

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

312 **Table 1 – Command Verb Requirements for CIM_ComputerSystem**

Command Verb	Requirement	Comments
Show	Shall	See 6.1.2.
Reset	May	See 6.1.3.
Set	May	See 6.1.4.
Start	Shall	See 6.1.5.
Stop	Shall	See 6.1.6.

313 6.1.1 Ordering of Results

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

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

317 6.1.2 Show

318 This section describes how to implement the `show` verb when applied to an instance of
 319 CIM_ComputerSystem. Implementations shall support the use of the `show` verb with
 320 CIM_ComputerSystem.

321 The `show` verb is used to display information about the instance of CIM_ComputerSystem and the
 322 referenced properties from the associated instance of CIM_AssociatedPowerManagementService.

323 6.1.2.1 Show a Single Instance

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

325 6.1.2.1.1 Command Form

```
326 show <CIM_ComputerSystem single instance>
```

327 6.1.2.1.2 CIM Requirements

328 See `CIM_ComputerSystem` in the “CIM Elements” section of the [Power State Management Profile](#) for the
329 list of mandatory properties.

330 6.1.2.1.3 Behavior Requirements

```
331 // the class definition for $instance includes two referenced properties,
332 // PowerState and PowerOnTime.
333 $instance=<CIM_ComputerSystem single instance>;
334 #Error=smOpReferences(
335     $instance->,
336     "CIM_AssociatedPowerManagementService",
337     NULL,
338     NULL,
339     {"PowerState", "PowerOnTime"},
340     $APMSInstancePaths[])
341 if (0 != #Error.code)
342 {
343     &smProcessOpError (#Error);
344     //includes &smEnd;
345 }
346 else
347 {
348 #propertynamelist[]={<array of mandatory non-key property names in CIM_ComputerSystem
349     (see CIM Requirements)>}
350 #additionalpropertylist[]={ "PowerState", "PowerOnTime" };
351 $APMSInstance = $APMSInstancePaths[1];
352 $instance.PowerState=$APMSInstance.PowerState;
353 $instance.PowerOnTime=$APMSInstance.PowerOnTime;
354 & smShowInstancePseudoProperties(
355     $instance,
356     #propertynamelist[],
357     #additionalpropertylist[]);
358 }
359 &smEnd;
```

360 6.1.2.2 Show Multiple Instances

361 This command form is for the `show` verb applied to multiple instances of `CIM_ComputerSystem`.

362 6.1.2.2.1 Command Form

```
363 show <CIM_ComputerSystem multiple instances>
```

364 **6.1.2.2.2 CIM Requirements**

365 See CIM_ComputerSystem in the “CIM Elements” section of the [Power State Management Profile](#) for the
366 list of mandatory properties.

367 **6.1.2.2.3 Behavior Requirements**368 **6.1.2.2.3.1 Preconditions**

369 \$containerInstance represents the instance of CIM_AdminDomain or CIM_ComputerSystem that is
370 associated to the targeted instances of CIM_ComputerSystem through the CIM_SystemComponent
371 association.

372 **6.1.2.2.3.2 Pseudo Code**

```

373 #Error=smOpAsociators(
374     $containerinstance->,
375     "CIM_SystemComponent",
376     NULL,
377     NULL,
378     NULL,
379     $CSInstancePaths[])
380 if (0 != #Error.code)
381 {
382     &smProcessOpError (#Error);
383     //includes &smEnd;
384 }
385 else
386 {
387     for #i < n //n is the number of ComputerSystem instances
388     {
389         #Error=smOpReferences(
390             $CSInstancePaths[i]->,
391             "CIM_AssociatedPowerManagementService",
392             NULL,
393             NULL,
394             {"PowerState", "PowerOnTime"},
395             $APMSInstancePaths[])
396         if (0 != #Error.code)
397         {
398             &smProcessOpError (#Error);
399             //includes &smEnd;
400         }
401         else
402         {
403             #propertynamelist[]={<array of mandatory non-key property names in
404             CIM_ComputerSystem (see CIM Requirements)>}
405             #additionalpropertylist[]={ "PowerState", "PowerOnTime" };
406             $APMSinstance = $APMSInstancePaths[1];
407             $instance.PowerState=$APMSinstance.PowerState;
408             $instance.PowerOnTime=$APMSinstance.PowerOnTime;

```

```

409     &smShowInstancePseudoProperties(
410         $CSIInstancePaths[i],
411         #propertynamelist[],
412         #additionalpropertylist[]);
413     }
414 }
415 }
416 &smEnd;

```

417 6.1.3 Reset

418 This section describes how to implement the `reset` verb when applied to an instance of
419 `CIM_ComputerSystem`. Implementations may support the use of the `reset` verb with
420 `CIM_ComputerSystem`.

421 6.1.3.1 Command Form

```
422 reset <CIM_ComputerSystem single instance>
```

423 6.1.3.2 CIM Requirements

```

424 uint32 RequestPowerStateChange(
425     uint16 PowerState,
426     CIM_ManagedElement REF ManagedElement,
427     datetime Time,
428     CIM_ConcreteJob REF Job,
429     datetime TimeoutPeriod);

```

430 6.1.3.3 Behavior Requirements

```

431 #Error = &smOpAssociators(
432     $instance.getObjectPath(),
433     "CIM_AssociatedPowerManagementService",
434     "CIM_PowerManagementService",
435     "UserOfService",
436     "ServiceProvided",
437     NULL,
438     $PMSInstancePaths[])
439 if (0 != #Error.code)
440 {
441     &smProcessOpError (#Error);
442     //includes &smEnd;
443 }
444 else if (PMSInstancePaths.length() > 0)
445 {
446     $PMSInstance = $PMSInstancePaths[1];
447     // 5 is equivalent to Power Cycle or System Reset;
448     smRequestPowerStateChange(5, instance->, NULL, PMSInstance->);
449 }

```

```

450 else
451     {
452         //unspecified return code, generic failure
453         $OperationError = smNewInstance("CIM_Error");
454         //CIM_ERR_FAILED
455         $OperationError.CIMStatusCode = 1;
456         //Other
457         $OperationError.ErrorType = 1;
458         //Low
459         $OperationError.PerceivedSeverity = 2;
460         $OperationError.OwningEntity = DMTF:SMCLP;
461         $OperationError.MessageID = 0x00000001;
462         $OperationError.Message = "Operation is not supported";
463         &smAddError($job, $OperationError);
464         &smMakeCommandStatus($job);
465     }
466 &smEnd;

```

467 6.1.4 Set

468 6.1.4.1 Set Command Form to Change the PowerState with PowerOnTime Property Value

469 This section describes how to implement the `set` verb that is used to set the power state of the computer
470 system. Implementations may support the use of the `set` verb with `CIM_ComputerSystem`.

471 The requirement for supporting modification of a property using this command form shall be equivalent to
472 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
473 in the [Power State Management Profile](#).

474 Time is entered either as a regular date-time value or as an interval value.

475 6.1.4.1.1 Command Form

```

476 set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
477     powerontime=<requestedtime>

```

478 6.1.4.1.2 CIM Requirements

```

479 uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
480 datetime CIM_AssociatedPowerManagementService.PowerOnTime;
481 uint32 RequestPowerStateChange(
482     uint16 PowerState,
483     CIM_ManagedElement REF ManagedElement,
484     datetime Time,
485     CIM_ConcreteJob REF Job,
486     datetime TimeoutPeriod);

```

487 6.1.4.1.3 Behavior Requirements

```

488 // The class definition for $instance includes two referenced properties,
489 // PowerState and PowerOnTime.
490 $instance=<CIM_ComputerSystem single instance>
491 #propertyName[] = {"PowerState"};
492 #propertyValues[] = {<powerstatevalue>};

```



```

493 #time = {<requestedtime>};
494 #Error = &smOpAssociators(
495     $instance.getObjectPath(),
496     "CIM_AssociatedPowerManagementService",
497     "CIM_PowerManagementService",
498     "UserOfService",
499     "ServiceProvided",
500     NULL,
501     $PMSInstancePaths[])
502 if (0 != #Error.code)
503     {
504     &smProcessOpError (#Error);
505     //includes &smEnd;
506     }
507 else
508     {
509     $PMSInstance = $PMSInstancePaths[1];
510     &smRequestPowerStateChange($PMSInstance, #propertyName[], #time,
511     #propertyValues[]);
512     }
513 #Error=smOpReferences(
514     $instance->,
515     "CIM_AssociatedPowerManagementService",
516     NULL,
517     NULL,
518     {"PowerState", "PowerOnTime"},
519     $APMSInstancePaths[])
520 if (0 != #Error.code)
521     {
522     &smProcessOpError (#Error);
523     //includes &smEnd;
524     }
525 else
526     {
527     #propertynamelist[]={<array of mandatory non-key property names in
528     CIM_ComputerSystem (see CIM Requirements)>}
529     #additionalpropertylist[]={ "PowerState", "PowerOnTime" };
530     $APMSInstance = $APMSInstancePaths[1];
531     $instance.PowerState=$APMSInstance.PowerState;
532     $instance.PowerOnTime=$APMSInstance.PowerOnTime;
533     &smShowInstancePseudoProperties(
534     $instance,
535     #propertynamelist[],
536     #additionalpropertylist[]);
537     }
538 &smEnd;

```

539 6.1.4.2 Set Command Form to Change the PowerState with No Time Property

540 This section describes how to implement the `set` verb that is used to set the power state of the computer
541 system. Implementations may support the use of the `set` verb with `CIM_ComputerSystem`.

542 The requirement for supporting modification of a property using this command form shall be equivalent to
543 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
544 in the [Power State Management Profile](#).

545 6.1.4.2.1 Command Form

```
546 set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
```

547 6.1.4.2.2 CIM Requirements

```
548 uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
549 uint32 RequestPowerStateChange(
550     uint16 PowerState,
551     CIM_ManagedElement REF ManagedElement,
552     datetime Time,
553     CIM_ConcreteJob REF Job,
554     datetime TimeoutPeriod);
```

555 6.1.4.2.3 Behavior Requirements

```
556 // The class definition for $instance includes one referenced property,
557 // PowerState.
558 $instance=<CIM_ComputerSystem single instance>
559 #propertyNames[] = {"PowerState"};
560 #propertyValues[] = {<powerstatevalue>};
561 #Error = &smOpAssociators(
562     $instance.getObjectPath(),
563     "CIM_AssociatedPowerManagementService",
564     "CIM_PowerManagementService",
565     "UserOfService",
566     "ServiceProvided",
567     NULL,
568     $PMSInstancePaths[])
569 if (0 != #Error.code)
570 {
571     &smProcessOpError (#Error);
572     //includes &smEnd;
573 }
574 else
575 {
576     $PMSInstance = $PMSInstancePaths[1];
577     &smRequestPowerStateChange($PMSInstance, #propertyNames[], NULL,
578     #propertyValues[]);
579 }
580 #Error=smOpReferences(
581     $instance->,
582     "CIM_AssociatedPowerManagementService",
583     NULL,
```

```

584     NULL,
585     {"PowerState"},
586     $APMSInstancePaths[])
587 if (0 != #Error.code)
588 {
589     &smProcessOpError (#Error);
590     //includes &smEnd;
591 }
592 else
593 {
594     #propertynamelist[]={<array of mandatory non-key property names in
595     CIM_ComputerSystem (see CIM Requirements)>}
596     #additionalpropertylist[]={ "PowerState", };
597     $APMSInstance = $APMSInstancePaths[1];
598     $instance.PowerState=$APMSInstance.PowerState;
599     &smShowInstancePseudoProperties(
600     $instance,
601     #propertynamelist[],
602     #additionalpropertylist[]);
603 }
604 &smEnd;

```

605 6.1.4.3 Set Command Form to Change the PowerState with #Time Value Format

606 This section describes how to implement the `set` verb that is used to set the power state of the computer
607 system. Implementations may support the use of the `set` verb with `CIM_ComputerSystem`.

608 The requirement for supporting modification of a property using this command form shall be equivalent to
609 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
610 in the [Power State Management Profile](#).

611 The property value format for `#Time` is defined in [DSP0216](#).

612 6.1.4.3.1 Command Form

```

613 set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
614     powerontime#time=<timestamp>

```

615 6.1.4.3.2 CIM Requirements

```

616 uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
617 datetime CIM_AssociatedPowerManagementService.PowerOnTime;
618 uint32 RequestPowerStateChange(
619     uint16 PowerState,
620     CIM_ManagedElement REF ManagedElement,
621     datetime Time,
622     CIM_ConcreteJob REF Job,
623     datetime TimeoutPeriod);

```

624 **6.1.4.3.3 Behavior Requirements**

```
625 // The class definition for $instance includes two referenced properties,
626 // PowerState and PowerOnTime.
627 $instance=<CIM_ComputerSystem single instance>
628 #propertyName[] = {"PowerState"};
629 #propertyValues[] = {<powerstatevalue>};
630 #time = {< timestamp converted into datetime datatype timestamp format>};
631 #Error = &smOpAssociators(
632     $instance.getObjectPath(),
633     "CIM_AssociatedPowerManagementService",
634     "CIM_PowerManagementService",
635     "UserOfService",
636     "ServiceProvided",
637     NULL,
638     $PMSInstancePaths[])
639
640 if (0 != #Error.code)
641     {
642     &smProcessOpError (#Error);
643     //includes &smEnd;
644     }
645 else
646     {
647     $PMSInstance = $PMSInstancePaths[1];
648     &smRequestPowerStateChange($PMSInstance, #propertyName[], #time,
649     #propertyValues[]);
650     }
651 #Error=smOpReferences(
652     $instance->,
653     "CIM_AssociatedPowerManagementService",
654     NULL,
655     NULL,
656     {"PowerState", "PowerOnTime"},
657     $APMSInstancePaths[])
658 if (0 != #Error.code)
659     {
660     &smProcessOpError (#Error);
661     //includes &smEnd;
662     }
663 else
664     {
665     #propertynamelist[]={<array of mandatory non-key property names in
666     CIM_ComputerSystem (see CIM Requirements)>}
667     #additionalpropertylist[]={ "PowerState", "PowerOnTime" };
668     $APMSInstance = $APMSInstancePaths[1];
669     $instance.PowerState=$APMSInstance.PowerState;
670     $instance.PowerOnTime=$APMSInstance.PowerOnTime;
```

```

671     &smShowInstancePseudoProperties(
672         $instance,
673         #propertynamelist[],
674         #additionalpropertylist[]);
675     }
676 &smEnd;

```

677 6.1.4.4 Set Command Form to Change the PowerState with #Interval Value Format

678 This section describes how to implement the `set` verb that is used to set the power state of the computer
679 system. Implementations may support the use of the `set` verb with `CIM_ComputerSystem`.

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

683 Time is entered in the user friendly time interval format. The property value format for `#Interval` is defined
684 in [DSP0216](#).

685 6.1.4.4.1 Command Form

```

686 set <CIM_ComputerSystem single instance> powerstate=<powerstatevalue>
687     powerontime#interval=<userfriendlyinterval>

```

688 6.1.4.4.2 CIM Requirements

```

689 uint16 CIM_AssociatedPowerManagementService.RequestedPowerState;
690 datetime CIM_AssociatedPowerManagementService.PowerOnTime;
691 uint32 RequestPowerStateChange(
692     uint16 PowerState,
693     CIM_ManagedElement REF ManagedElement,
694     datetime Time,
695     CIM_ConcreteJob REF Job,
696     datetime TimeoutPeriod);

```

697 6.1.4.4.3 Behavior Requirements

```

698 // the class definition for $instance includes two referenced properties,
699 // PowerState and PowerOnTime.
700 $instance=<CIM_ComputerSystem single instance>
701 #propertyName[] = {"PowerState"};
702 #propertyValues[] = {<powerstatevalue>};
703 #time = {< userfriendlyinterval converted into datetime datatype interval format>};
704 #Error = &smOpAssociators(
705     $instance.getObjectPath(),
706     "CIM_AssociatedPowerManagementService",
707     "CIM_PowerManagementService",
708     "UserOfService",
709     "ServiceProvided",
710     NULL,
711     $PMSInstancePaths[])

```

```

712 if (0 != #Error.code)
713     {
714         &smProcessOpError (#Error);
715         //includes &smEnd;
716     }
717 else
718     {
719         $PMSinstance = $PMSInstancePaths[1];
720         &smRequestPowerStateChange($PMSinstance, #propertyName[], #time,
721         #propertyValues[]);
722     }
723     #Error=smOpReferences(
724         $instance->,
725         "CIM_AssociatedPowerManagementService",
726         NULL,
727         NULL,
728         {"PowerState", "PowerOnTime"},
729         $APMSInstancePaths[])
730 if (0 != #Error.code)
731     {
732         &smProcessOpError (#Error);
733         //includes &smEnd;
734     }
735 else
736     {
737         #propertynamelist[]={<array of mandatory non-key property names in
738         CIM_ComputerSystem (see CIM Requirements)>}
739         #additionalpropertylist[]={ "PowerState", "PowerOnTime" };
740         $APMSinstance = $APMSInstancePaths[1];
741         $instance.PowerState=$APMSinstance.PowerState;
742         $instance.PowerOnTime=$APMSinstance.PowerOnTime;
743         &smShowInstancePseudoProperties(
744             $instance,
745             #propertynamelist[],
746             #additionalpropertylist[]);
747     }
748 &smEnd;

```

749 6.1.5 Start

750 This section describes how to implement the `start` verb when applied to an instance of
751 `CIM_ComputerSystem`. Implementations may support the use of the `start` verb with
752 `CIM_ComputerSystem`.

753 The `start` verb is used to enable a computer system.

754 6.1.5.1 Command Form

```
755 start < CIM_ComputerSystem single instance >
```

756 **6.1.5.2 CIM Requirements**

```

757 uint32 RequestPowerStateChange(
758     uint16 PowerState,
759     CIM_ManagedElement REF ManagedElement,
760     datetime Time,
761     CIM_ConcreteJob REF Job,
762     datetime TimeoutPeriod);

```

763 **6.1.5.3 Behavior Requirements**

```

764 $instance=<CIM_ComputerSystem single instance>;
765 #Error = &smOpAssociators(
766     $instance.getObjectPath(),
767     "CIM_AssociatedPowerManagementService",
768     "CIM_PowerManagementService",
769     "UserOfService",
770     "ServiceProvided",
771     NULL,
772     $PMSInstancePaths[])
773 if (0 != #Error.code)
774     {
775     &smProcessOpError (#Error);
776     //includes &smEnd;
777     }
778 else if (PMSInstancePaths.length() > 0)
779     {
780     $PMSInstance = $PMSInstancePaths[1];
781     // 2 is equivalent to Power On;
782     smRequestPowerStateChange(2, instance->, NULL,PMSInstance->);
783     }
784 else
785     {
786     //unspecified return code, generic failure
787     $OperationError = smNewInstance("CIM_Error");
788     //CIM_ERR_FAILED
789     $OperationError.CIMStatusCode = 1;
790     //Other
791     $OperationError.ErrorType = 1;
792     //Low
793     $OperationError.PerceivedSeverity = 2;
794     $OperationError.OwningEntity = DMTF:SMCLP;
795     $OperationError.MessageID = 0x00000001;
796     $OperationError.Message = "Operation is not supported";
797     &smAddError($job, $OperationError);
798     &smMakeCommandStatus($job);
799     }
800 &smEnd;

```

801 **6.1.6 Stop**

802 This section describes how to implement the `stop` verb when applied to an instance of
 803 `CIM_ComputerSystem`. Implementations may support the use of the `stop` verb with
 804 `CIM_ComputerSystem`.

805 The `stop` verb is used to disable a computer system.

806 **6.1.6.1 Command Form**

```
807 stop < CIM_ComputerSystem single instance >
```

808 **6.1.6.2 CIM Requirements**

```
809 uint32 RequestPowerStateChange(  
810     uint16 PowerState,  
811     CIM_ManagedElement REF ManagedElement,  
812     datetime Time,  
813     CIM_ConcreteJob REF Job,  
814     datetime TimeoutPeriod);
```

815 **6.1.6.3 Behavior Requirements**

```
816 $instance=<CIM_ComputerSystem single instance>;  
817 #Error = &smOpAssociators(  
818     $instance.getObjectPath(),  
819     "CIM_AssociatedPowerManagementService",  
820     "CIM_PowerManagementService",  
821     "UserOfService",  
822     "ServiceProvided",  
823     NULL,  
824     $PMSInstancePaths[])  
825 if (0 != #Error.code)  
826     {  
827         &smProcessOpError (#Error);  
828         //includes &smEnd;  
829     }  
830 else if (PMSInstancePaths.length() > 0)  
831     {  
832         $PMSInstance = $PMSInstancePaths[1];  
833         // 8 is equivalent to Power Off(Soft)  
834         smRequestPowerStateChange(8, instance->, NULL, PMSInstance->);  
835     }  
836 else  
837     {  
838         //unspecified return code, generic failure  
839         $OperationError = smNewInstance("CIM_Error");  
840         //CIM_ERR_FAILED  
841         $OperationError.CIMStatusCode = 1;  
842         //Other  
843         $OperationError.ErrorType = 1;  
844         //Low  
845         $OperationError.PerceivedSeverity = 2;  
846         $OperationError.OwningEntity = DMTF:SMCLP;  
847         $OperationError.MessageID = 0x00000001;
```



```

848     $OperationError.Message = "Operation is not supported";
849     &smAddError($job, $OperationError);
850     &smMakeCommandStatus($job);
851 }
852 &smEnd;
    
```

853 **6.2 CIM_PowerManagementService**

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

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

860 **Table 2 – Command Verb Requirements for CIM_PowerManagementService**

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	

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

862 **6.2.1 Ordering of Results**

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

- 865 • Results for `CIM_PowerManagementService` are unordered; therefore, no algorithm is defined.

866 **6.2.2 Show**

867 This section describes how to implement the `show` verb when applied to an instance of
 868 `CIM_PowerManagementService`. Implementations shall support the use of the `show` verb with
 869 `CIM_PowerManagementService`.

870 The `show` verb is used to display information about `CIM_PowerManagementService` instances.

871 **6.2.2.1 Show Command Form for Single Instance Target**

872 **6.2.2.1.1 Command Form**

```

873 show <CIM_PowerManagementService single instance>
    
```

874 **6.2.2.1.2 CIM Requirements**

875 See CIM_PowerManagementService in the “CIM Elements” section of the [Power State Management Profile](#) for the list of mandatory properties.

877 **6.2.2.1.3 Behavior Requirements**878 **6.2.2.1.3.1 Preconditions**

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

880 **6.2.2.1.3.2 Pseudo Code**

```
881 $instance=<CIM_PowerManagementService single instance>;
882 #propertylist[] = NULL;
883 if ( false == #all )
884 {
885     #propertylist[] = {<array of mandatory non-key property names (see CIM
886         Requirements)>}
887 }
888 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
889 &smEnd;
```

890 **6.2.2.2 Show Command Form for Multiple Instances Target**891 **6.2.2.2.1 Command Form**

```
892 show <CIM_PowerManagementService multiple instances>
```

893 **6.2.2.2.2 CIM Requirements**

894 See CIM_PowerManagementService in the “CIM Elements” section of the [Power State Management Profile](#) for the list of mandatory properties.

896 **6.2.2.2.3 Behavior Requirements**897 **6.2.2.2.3.1 Preconditions**

898 \$containerInstance contains the instance of CIM_ComputerSystem that is associated to the targeted instances of CIM_PowerManagementServices through the CIM_HostedService association.

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

901 **6.2.2.2.3.2 Pseudo Code**

```
902 #propertylist[] = NULL;
903 if ( false == #all )
904 {
905     #propertylist[] = {<array of mandatory non-key property names (see CIM
906         Requirements)>}
907 }
908 &smShowInstances ( "CIM_PowerManagementService", "CIM_HostedService",
909     $containerInstance.getObjectPath(), #propertylist[] );
910 &smEnd;
```

911 6.3 CIM_PowerManagementCapabilities

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

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

918 **Table 3 – Command Verb Requirements for CIM_PowerManagementCapabilities**

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	

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

921 6.3.1 Ordering of Results

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

- 924 • Results for `CIM_PowerManagementCapabilities` are unordered; therefore, no algorithm is
 925 defined.

926 6.3.2 Show

927 This section describes how to implement the `show` verb when applied to an instance of
 928 `CIM_PowerManagementCapabilities`. Implementations shall support the use of the `show` verb with
 929 `CIM_PowerManagementCapabilities`.

930 The `show` verb is used to display information about `CIM_PowerManagementCapabilities` instances.

931 6.3.2.1 Show Command Form for a Single Instance Target

932 6.3.2.1.1 Command Form

933 `show < CIM_PowerManagementCapabilities single instance >`

934 6.3.2.1.2 CIM Requirements

935 See `CIM_PowerManagementCapabilities` in the “CIM Elements” section of the [Power State Management](#)
 936 [Profile](#) for the list of mandatory properties.

937 **6.3.2.1.3 Behavior Requirements**938 **6.3.2.1.3.1 Preconditions**

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

940 **6.3.2.1.3.2 Pseudo Code**

```
941 $instance=<CIM_PowerManagementCapabilities single instance>;
942 #propertylist[] = NULL;
943 if ( false == #all )
944     {
945         #propertylist[] = {<array of mandatory non-key property names (see CIM
946             Requirements)>}
947     }
948 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
949 &smEnd;
```

950 **6.3.2.2 Show Command Form for Multiple Instances Target**951 **6.3.2.2.1 Command Form**

```
952 show < CIM_PowerManagementCapabilities multiple instances>
```

953 **6.3.2.2.2 CIM Requirements**

954 See CIM_PowerManagementCapabilities in the “CIM Elements” section of the [Power State Management Profile](#) for the list of mandatory properties.

956 **6.3.2.2.3 Behavior Requirements**957 **6.3.2.2.3.1 Preconditions**

958 \$containerInstance represents the instance of CIM_ConcreteCollection with ElementName property
959 that contains “Capabilities” and is associated to the targeted instances of
960 CIM_PowerManagementCapabilities through the CIM_MemberOfCollection association.

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

962 **6.3.2.2.3.2 Pseudo Code**

```
963 #propertylist[] = NULL;
964 if ( false == #all )
965     {
966         #propertylist[] = {<array of mandatory non-key property names (see CIM
967             Requirements)>}
968     }
969 &smShowInstances ( "CIM_PowerManagementCapabilities", "CIM_MemberOfCollection",
970     $containerInstance.getObjectPath(), #propertylist[] );
971 &smEnd;
```

972 **6.4 CIM_AssociatedPowerManagementService**

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

974 Table 4 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
975 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and

976 target. Table 4 is for informational purposes only; in case of a conflict between Table 4 and requirements
 977 detailed in the following sections, the text detailed in the following sections supersedes the information in
 978 Table 4.

979 **Table 4 – Command Verb Requirements for CIM_AssociatedPowerManagementService**

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	

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

982 **6.4.1 Ordering of Results**

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

- 985 • Results for CIM_AssociatedPowerManagementService are unordered; therefore, no algorithm
 986 is defined.

987 **6.4.2 Show**

988 This section describes how to implement the `show` verb when applied to an instance of
 989 CIM_AssociatedPowerManagementService. Implementations shall support the use of the `show` verb with
 990 CIM_AssociatedPowerManagementService.

991 **6.4.2.1 Show Command Form for a Single Instance Target – CIM_ComputerSystem Reference**

992 This command form is used to show a single instance of CIM_AssociatedPowerManagementService.
 993 This command form corresponds to a `show` command issued against instances of
 994 CIM_AssociatedPowerManagementService where only one reference is specified and the reference is to
 995 the scoping instance of CIM_ComputerSystem.

996 **6.4.2.1.1 Command Form**

997 `show <CIM_AssociatedPowerManagementService single instance>`

998 **6.4.2.1.2 CIM Requirements**

999 See CIM_AssociatedPowerManagementService in the “CIM Elements” section of the [Power State](#)
 1000 [Management Profile](#) for the list of mandatory properties.

1001 **6.4.2.1.3 Behavior Requirements**1002 **6.4.2.1.3.1 Preconditions**

1003 \$instance represents the instance of a CIM_ComputerSystem, which is referenced by
1004 CIM_AssociatedPowerManagementService.

1005 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1006 **6.4.2.1.3.2 Pseudo Code**

```
1007 $instance=<CIM_ComputerSystem single instance>;
1008 #propertylist[] = NULL;
1009 if ( false == #all )
1010 {
1011     #propertylist[] = <array of mandatory non-key property names (see CIM
1012     Requirements)>;
1013 }
1014 &smShowAssociationInstances ( "CIM_AssociatedPowerManagementService",
1015     $instance.getObjectPath(), #propertylist[] );
1016 &smEnd;
```

1017 **6.4.2.2 Show Command Form for Multiple Instances Target – CIM_PowerManagementService Reference**
1018

1019 This command form is used to show many instances of CIM_AssociatedPowerManagementService. This
1020 command form corresponds to a show command issued against instances of
1021 CIM_AssociatedPowerManagementService where only one reference is specified and the reference is to
1022 the scoping instance of CIM_PowerManagementService.

1023 **6.4.2.2.1 Command Form**

```
1024 show <CIM_AssociatedPowerManagementService multiple instances>
```

1025 **6.4.2.2.2 CIM Requirements**

1026 See CIM_AssociatedPowerManagementService in the “CIM Elements” section of the [Power State Management Profile](#) for the list of mandatory properties.

1028 **6.4.2.2.3 Behavior Requirements**1029 **6.4.2.2.3.1 Preconditions**

1030 \$instance represents the instance of a CIM_PowerManagementService, which is referenced by
1031 CIM_AssociatedPowerManagementService.

1032 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1033 **6.4.2.2.3.2 Pseudo Code**

```
1034 $instance=<CIM_PowerManagementService single instance>;
1035 #propertylist[] = NULL;
1036 if ( false == #all )
1037 {
1038     #propertylist[] = <array of mandatory non-key property names (see CIM
1039     Requirements)>;
1040 }
```

```

1041 &smShowAssociationInstances ( "CIM_AssociatedPowerManagementService",
1042     $instance.getObjectPath(), #propertylist[] );
1043 &smEnd;

```

1044 6.4.2.3 Show Command Form for a Single Instance Target – Both References

1045 This command form is for the show verb applied to a single instance. This command form corresponds to
 1046 a show command issued against CIM_AssociatedPowerManagementService where both references are
 1047 specified and therefore the desired instance is unambiguously identified.

1048 6.4.2.3.1 Command Form

```

1049 show <CIM_AssociatedPowerManagementService single instance>

```

1050 6.4.2.3.2 CIM Requirements

1051 See CIM_AssociatedPowerManagementService in the “CIM Elements” section of the [Power State](#)
 1052 [Management Profile](#) for the list of mandatory properties.

1053 6.4.2.3.3 Behavior Requirements

1054 6.4.2.3.3.1 Preconditions

1055 \$instance represents the instanceA of a CIM_ComputerSystem, which is referenced by
 1056 CIM_AssociatedPowerManagementService.

1057 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1058 6.4.2.3.3.2 Pseudo Code

```

1059 $instanceA=<CIM_ComputerSystem single instance>;
1060 $instanceB=<CIM_PowerManagementService single instance>;
1061 #propertylist[] = NULL;
1062 if ( false == #all )
1063     {
1064         #propertylist[] = <array of mandatory non-key property names (see CIM
1065             Requirements)>;
1066     }
1067 &smShowAssociationInstance ( "CIM_AssociatedPowerManagementService",
1068     $instanceA.getObjectPath(), $instanceB.getObjectPath(), #propertylist[] );
1069 &smEnd;

```

1070 6.5 CIM_ElementCapabilities

1071 The cd, help, version, and exit verbs shall be supported as described in [DSP0216](#).

1072 Table 5 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1073 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1074 target. Table 5 is for informational purposes only; in case of a conflict between Table 5 and requirements
 1075 detailed in the following sections, the text detailed in the following sections supersedes the information in
 1076 Table 5.

1077

Table 5 – 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 6.5.2.
Start	Not supported	
Stop	Not supported	

1078 No mapping is defined for the following verbs for the specified target: `create`, `delete`, `dump`, `load`,
1079 `reset`, `set`, `start`, and `stop`.

1080 6.5.1 Ordering of Results

1081 When results are returned for multiple instances of `CIM_ElementCapabilities`, implementations shall
1082 utilize the following algorithm to produce the natural (that is, default) ordering:

- 1083 • Results for `CIM_ElementCapabilities` are unordered; therefore, no algorithm is defined.

1084 6.5.2 Show

1085 This section describes how to implement the `show` verb when applied to an instance of
1086 `CIM_ElementCapabilities`. Implementations shall support the use of the `show` verb with
1087 `CIM_ElementCapabilities`.

1088 6.5.2.1 Show Command Form for Multiple Instances Target – `CIM_PowerManagementService` 1089 Reference

1090 This command form is used to show many instances of `CIM_ElementCapabilities`. This command form
1091 corresponds to a `show` command issued against instances of `CIM_ElementCapabilities` where only one
1092 reference is specified and the reference is to the scoping instance of `CIM_PowerManagementService`.

1093 6.5.2.1.1 Command Form

```
1094 show <CIM_ElementCapabilities multiple instances>
```

1095 6.5.2.1.2 CIM Requirements

1096 See `CIM_ElementCapabilities` in the “CIM Elements” section of the [Power State Management Profile](#) for
1097 the list of mandatory properties.

1098 6.5.2.1.2.1 Behavior Requirements

1099 6.5.2.1.2.2 Preconditions

1100 `$instance` represents the instance of a `CIM_PowerManagementService`, which is referenced by
1101 `CIM_ElementCapabilities`.

1102 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1103 6.5.2.1.2.3 Pseudo Code

```

1104 $instance=<CIM_PowerManagementService single instance>;
1105 #propertylist[] = NULL;
1106 if ( false == #all )
1107     {
1108         #propertylist[] = <array of mandatory non-key property names (see CIM
1109             Requirements)>;
1110     }
1111 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),
1112     #propertylist[] );
1113 &smEnd;

```

1114 6.5.2.2 Show Command Form for Multiple Instances – CIM_PowerManagementCapabilities 1115 Reference

1116 This command form is used to show multiple instances of CIM_ElementCapabilities. This command form
1117 corresponds to a `show` command issued against multiple instances of CIM_ElementCapabilities where
1118 only one reference is specified and the reference is to the scoping instance of
1119 CIM_PowerManagementCapabilities.

1120 6.5.2.2.1 Command Form

```
1121 show <CIM_ElementCapabilities multiple instances>
```

1122 6.5.2.2.2 CIM Requirements

1123 See CIM_ElementCapabilities in the “CIM Elements” section of the [Power State Management Profile](#) for
1124 the list of mandatory properties.

1125 6.5.2.2.3 Behavior Requirements

1126 6.5.2.2.3.1 Preconditions

1127 In this section `$instance` represents the instance of a CIM_PowerManagementCapabilities, which is
1128 referenced by CIM_ElementCapabilities.

1129 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1130 6.5.2.2.3.2 Pseudo Code

```

1131 $instance=<CIM_PowerManagementCapabilities single instance>;
1132 #propertylist[] = NULL;
1133 if ( false == #all )
1134     {
1135         #propertylist[] = <array of mandatory non-key property names (see CIM
1136             Requirements)>;
1137     }
1138 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath(),
1139     #propertylist[] );
1140 &smEnd;

```

1141 **6.5.2.3 Show Command Form for Single Instance Target – CIM_PowerManagementService and**
 1142 **CIM_PowerManagementCapabilities References**

1143 This command form is for the show verb applied to a single instance. This command form corresponds to
 1144 a show command issued against CIM_ElementCapabilities where both references are specified and
 1145 therefore the desired instance is unambiguously identified.

1146 **6.5.2.3.1 Command Form**

```
1147 show <CIM_ElementCapabilities single instance>
```

1148 **6.5.2.3.2 CIM Requirements**

1149 See CIM_ElementCapabilities in the “CIM Elements” section of the [Power State Management Profile](#) for
 1150 the list of mandatory properties.

1151 **6.5.2.3.3 Behavior Requirements**

1152 **6.5.2.3.3.1 Preconditions**

1153 \$instanceA represents the instance of a CIM_PowerManagementService and \$instanceB represents
 1154 the instance of a CIM_PowerManagementCapabilities, both of which are referenced by
 1155 CIM_ElementCapabilities.

1156 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1157 **6.5.2.3.3.2 Pseudo Code**

```
1158 $instanceA=<CIM_PowerManagementService single instance>;
1159 $instanceB=<CIM_PowerManagementCapabilities single instance>;
1160 #propertylist[] = NULL;
1161 if ( false == #all )
1162 {
1163     #propertylist[] = <array of mandatory non-key property names (see CIM
1164     Requirements)>;
1165 }
1166 &smShowAssociationInstance ("CIM_ElementCapabilities", $instanceA.getObjectPath(),
1167     $instanceB.getObjectPath(), #propertylist[] );
1168 &smEnd;
```

1169 **6.6 CIM_HostedService**

1170 The cd, help, version, and exit verbs shall be supported as described in [DSP0216](#).

1171 Table 6 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
 1172 class, and when appropriate, a cross-reference to the section detailing the mapping for the verb and
 1173 target. Table 6 is for informational purposes only; in case of a conflict between Table 6 and requirements
 1174 detailed in the following sections, the text detailed in the following sections supersedes the information in
 1175 Table 6.

1176

Table 6 – 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 6.6.2.
Start	Not supported	
Stop	Not supported	

1177 No mapping is defined for the following verbs for the specified target: *create*, *delete*, *dump*, *load*,
 1178 *reset*, *set*, *start*, and *stop*.

1179 **6.6.1 Ordering of Results**

1180 When results are returned for multiple instances of CIM_HostedService, implementations shall utilize the
 1181 following algorithm to produce the natural (that is, default) ordering:

- 1182 • Results for CIM_HostedService are unordered; therefore, no algorithm is defined.

1183 **6.6.2 Show**

1184 This section describes how to implement the *show* verb when applied to an instance of
 1185 CIM_HostedService. Implementations shall support the use of the *show* verb with CIM_HostedService.

1186 **6.6.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference**

1187 This command form is used to show many instances of CIM_HostedService. This command form
 1188 corresponds to a *show* command issued against instances of CIM_HostedService where only one
 1189 reference is specified and the reference is to the scoping instance of CIM_ComputerSystem.

1190 **6.6.2.1.1 Command Form**

1191 `show <CIM_HostedService multiple instances>`

1192 **6.6.2.1.2 CIM Requirements**

1193 See CIM_HostedService in the “CIM Elements” section of the [Power State Management Profile](#) for the list
 1194 of mandatory properties.

1195 **6.6.2.1.3 Behavior Requirements**

1196 **6.6.2.1.3.1 Preconditions**

1197 *\$instance* represents the instance of a CIM_ComputerSystem, which is referenced by
 1198 CIM_HostedService.

1199 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1200 **6.6.2.1.3.2 Pseudo Code**

```

1201 $instance=<CIM_ComputerSystem single instance>;
1202 #propertylist[] = NULL;
1203 if ( false == #all )
1204     {
1205         #propertylist[] = <array of mandatory non-key property names (see CIM
1206             Requirements)>;
1207     }
1208 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath(),
1209     #propertylist[] );
1210 &smEnd;

```

1211 **6.6.2.2 Show Command Form for a Single Instance – CIM_PowerManagementService Reference**

1212 This command form is used to show a single instance of CIM_HostedService. This command form
 1213 corresponds to a `show` command issued against a single instance of CIM_HostedService where only one
 1214 reference is specified and the reference is to the instance of CIM_PowerManagementService

1215 **6.6.2.2.1 Command Form**

```
1216 show <CIM_HostedService single instance>
```

1217 **6.6.2.2.2 CIM Requirements**

1218 See CIM_HostedService in the “CIM Elements” section of the [Power State Management Profile](#) for the list
 1219 of mandatory properties.

1220 **6.6.2.2.3 Behavior Requirements**1221 **6.6.2.2.3.1 Preconditions**

1222 In this section `$instance` represents the instance of a CIM_PowerManagementService, which is
 1223 referenced by CIM_HostedService.

1224 `#all` is true if the “-all” option was specified with the command; otherwise, `#all` is false.

1225 **6.6.2.2.3.2 Pseudo Code**

```

1226 $instance=<CIM_PowerManagementService single instance>
1227 #propertylist[] = NULL;
1228 if ( false == #all )
1229     {
1230         #propertylist[] = <array of mandatory non-key property names (see CIM
1231             Requirements)>;
1232     }
1233 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath(),
1234     #propertylist[] );
1235 &smEnd;

```

1236 **6.6.2.3 Show Command Form for a Single Instance Target – Both References**

1237 This command form is for the `show` verb applied to a single instance. This command form corresponds to
 1238 a `show` command issued against CIM_HostedService where both references are specified and therefore
 1239 the desired instance is unambiguously identified.

1240 **6.6.2.3.1 Command Form**1241 `show <CIM_HostedService single instance>`1242 **6.6.2.3.2 CIM Requirements**

1243 See CIM_HostedService in the “CIM Elements” section of the [Power State Management Profile](#) for the list
 1244 of mandatory properties.

1245 **6.6.2.3.3 Behavior Requirements**1246 **6.6.2.3.3.1 Preconditions**

1247 \$instanceA represents the instance of a CIM_ComputerSystem and \$instanceB represents the
 1248 instance of CIM_PowerManagementservice, both of which are referenced by CIM_HostedService.

1249 #all is true if the “-all” option was specified with the command; otherwise, #all is false.

1250 **6.6.2.3.3.2 Pseudo Code**

```

1251 $instanceA=<CIM_ComputerSystem single instance>;
1252 $instanceB=<CIM_PowerManagementService single instance>;
1253 #propertylist[] = NULL;
1254 if ( false == #all )
1255     {
1256         #propertylist[] = <array of mandatory non-key property names (see CIM
1257             Requirements)>;
1258     }
1259 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),
1260     $instanceB.getObjectPath(), #propertylist[]);
1261 &smEnd;

```

1262

ANNEX A
(informative)

Change Log

1263
1264
1265
1266
1267

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

1268