



1
2
3
4

Document Number: DSP0834

Date: 2009-06-04

Version: 1.0.0

5 **Computer System Profile SM CLP Command**
6 **Mapping Specification**

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

10

11 Copyright notice

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

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

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27 withdrawn or modified after publication, and shall be indemnified and held harmless by any party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

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

33

CONTENTS

34	Foreword	5
35	Introduction	6
36	1 Scope	7
37	2 Normative References.....	7
38	2.1 Approved References	7
39	2.2 Other References.....	7
40	3 Terms and Definitions.....	7
41	4 Symbols and Abbreviated Terms.....	8
42	5 Recipes.....	9
43	5.1 IAddReferencedProperties.....	9
44	6 Mappings.....	11
45	6.1 CIM_ComputerSystem.....	11
46	6.2 CIM_ElementCapabilities	20
47	6.3 CIM_EnabledLogicalElementCapabilities.....	23
48	6.4 CIM_HostedService	25
49	6.5 CIM_ServiceAffectsElement	27
50	6.6 CIM_TimeService	30
51	ANNEX A (informative) Change Log	32
52		

53 Tables

54	Table 1 – Local Recipes.....	9
55	Table 2 – Command Verb Requirements for CIM_ComputerSystem	12
56	Table 3 – Command Verb Requirements for CIM_ElementCapabilities	21
57	Table 4 – Command Verb Requirements for CIM_EnabledLogicalElementCapabilities.....	23
58	Table 5 – Command Verb Requirements for CIM_HostedService	25
59	Table 6 – Command Verb Requirements for CIM_ServiceAffectsElement	27
60	Table 7 – Command Verb Requirements for CIM_TimeService	30
61		

63

Foreword

64 The *Computer System Profile SM CLP Command Mapping Specification* (DSP0834) was prepared by the
65 DMTF 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 DMTF Server Management Working
71 Group:

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

79

80

Introduction

81 This document defines the SM CLP mapping for CIM elements described in the [Computer System Profile](#).
82 The information in this specification, combined with the *SM CLP-to-CIM Common Mapping*
83 *Specification 1.0* ([DSP0216](#)), is intended to be sufficient to implement SM CLP commands relevant to the
84 classes, properties, and methods described in the [Computer System Profile](#) using CIM operations.

85 The target audience for this specification is implementers of the SM CLP support for the [Computer](#)
86 [System Profile](#).

87 **Computer System Profile SM CLP Command Mapping** 88 **Specification**

89 **1 Scope**

90 This specification contains the requirements for an implementation of the SM CLP to provide access to,
91 and implement the behaviors of, the [Computer System Profile](#).

92 **2 Normative References**

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

96 **2.1 Approved References**

97 DMTF DSP1052, *Computer System Profile 1.0*,
98 http://www.dmtf.org/standards/published_documents/DSP1052_1.0.pdf

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

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

103 **2.2 Other References**

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

106 **3 Terms and Definitions**

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

108 **3.1**

109 **can**

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

111 **3.2**

112 **cannot**

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

114 **3.3**

115 **conditional**

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

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

146 **4 Symbols and Abbreviated Terms**

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

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

- 157 **4.4**
 158 **SM**
 159 Server Management
- 160 **4.5**
 161 **SMI-S**
 162 Storage Management Initiative Specification
- 163 **4.6**
 164 **SNIA**
 165 Storage Networking Industry Association
- 166 **4.7**
 167 **UFsT**
 168 User Friendly selection Tag

169 **5 Recipes**

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

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

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

181

Table 1 – Local Recipes

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

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

183 **5.1 IAddReferencedProperties**

184 **5.1.1 Description**

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

186 **5.1.2 Pseudo Code**

```
187 // $device contains the instance of CIM_LogicalDevice to which associated properties
188 // should be added
189 // #all indicates whether or not the all option was specified for the command
190 sub IAddReferencedProperties($instance, #ReferencedPropertyNames[]) {
```

```
191     #propertylist[] = NULL;
192 // find the associated service, call the method as a query, insert the time as a
193 // referenced property
194 // Find the TimeService associated to the target system
195 #Error = smOpAssociators(
196     $instance.getObjectPath(),
197     "CIM_ServiceAffectsElement",
198     "CIM_TimeService",
199     NULL,
200     NULL,
201     $Services[]);
202 //if there isn't a service associated the function is not supported.
203 if (0 == $Services[].length){
204     //not supported, don't add property
205     return;
206 }
207 $Service-> = $Services[0].getObjectPath();
208 %InArguments[] = { newArgument("GetRequest", TRUE)
209     newArgument("ManagedElement", $instance.getOjectPath())}
210 %OutArguments[] = { newArgument("TimeData", #timedata)};
211 //invoke method
212 #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
213     "ManageTime",
214     %InArguments[],
215     %OutArguments[]);
216 // process return code to CLP Command Status
217 if (0 != #Error.code) {
218     //method invocation failed
219     if ( (null != #Error.$error) && (null != #Error.$error[0]) ) {
220         // if the method invocation contains an embedded error
221         // use it for the Error for the overall job
222         &smAddError($job, #Error.$error[0]);
223         &smMakeCommandStatus($job);
224         &smEnd;
225     }
226     else if (#Error.code == 17) {
227         //not supported, so don't add property
228         return;
229     }
230     else {
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     $instance.CurrentTime = #timedata;
251     #ReferencedPropertyNames = {"CurrentTime"};
252 }
253 else if (1 == #returnStatus) {
254     //not supported, so don't add
255     return;
256 }
257 else {
258     //generic failure
259     $OperationError = smNewInstance("CIM_Error");
260     //CIM_ERR_FAILED
261     $OperationError.CIMStatusCode = 1;
262     //Other
263     $OperationError.ErrorType = 1;
264     //Low
265     $OperationError.PerceivedSeverity = 2;
266     $OperationError.OwningEntity = DMTF:SMCLP;
267     $OperationError.MessageID = 0x00000002;
268     $OperationError.Message = "Failed. No further information is available.";
269     &smAddError($job, $OperationError);
270     &smMakeCommandStatus($job);
271 }
272 } //lAddReferencedProperties()

```

273 6 Mappings

274 The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in
 275 [Computer System Profile](#). Requirements specified here related to the support for a CLP verb for a
 276 particular class are solely within the context of this profile.

277 6.1 CIM_ComputerSystem

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

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

284

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.

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

286 6.1.1 Ordering of Results

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

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

290 6.1.2 Reset

291 This section describes how to implement the `reset` verb when applied to an instance of
292 CIM_ComputerSystem. Implementations may support the use of the `reset` verb with
293 CIM_ComputerSystem.

294 6.1.2.1 Command Form

```
295 reset <CIM_ComputerSystem single instance>
```

296 6.1.2.2 CIM Requirements

```
297 uint16 EnabledState;  
298 uint16 RequestedState;  
299 uint32 CIM_ComputerSystem.RequestStateChange (  
300     [IN] uint16 RequestedState,  
301     [OUT] REF CIM_ConcreteJob Job,  
302     [IN] datetime TimeoutPeriod );
```

303 6.1.2.3 Behavior Requirements

304 6.1.2.3.1 Preconditions

305 In this section `$instance` represents the targeted instance of CIM_ComputerSystem.

```
306 $instance=<CIM_ComputerSystem single instance>;
```

307 6.1.2.3.1.1 Pseudo Code

```
308 &smResetRSC ( $instance.getObjectPath() );  
309 &smEnd;
```

310 **6.1.3 Set**

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

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

315 **6.1.3.1 General Usage of Set for a Single Property**

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

318 The requirement for supporting modification of a property using this command form shall be equivalent to
 319 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
 320 in the [Computer System Profile](#).

321 **6.1.3.1.1 Command Form**

```
322 set <CIM_ComputerSystem single object> <propertyname>=<propertyvalue>
```

323 **6.1.3.1.2 CIM Requirements**

324 See `CIM_ComputerSystem` in the “CIM Elements” section of the [Computer System Profile](#) for the list of
 325 mandatory properties.

326 **6.1.3.1.3 Behavior Requirements**

```
327 $instance=<CIM_ComputerSystem single object>
328 #propertyName[] = {<propertyname>};
329 #propertyValues[] = {<propertyvalue>};
330 &smSetInstance ( $instance, #propertyName[], #propertyValues[] );
331 &smEnd;
```

332 **6.1.3.2 General Usage of Set for Multiple Properties**

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

336 The requirement for supporting modification of a property using this command form shall be equivalent to
 337 the requirement for supporting modification of the property using the `ModifyInstance` operation as defined
 338 in the [Computer System Profile](#).

339 **6.1.3.2.1 Command Form**

```
340 set <CIM_ComputerSystem multiple objects> <propertyname1>=<propertyvalue1>
341 <propertynamen>=<propertyvaluen>
```

342 **6.1.3.2.2 CIM Requirements**

343 See `CIM_ComputerSystem` in the “CIM Elements” section of the [Computer System Profile](#) for the list of
 344 mandatory properties.

345 6.1.3.2.3 Behavior Requirements

```

346 $instance=<CIM_ComputerSystem multiple objects>
347 #propertyNames[] = {<propertyname>};
348 for #i < n
349     {
350         #propertyNames[#i] = <propertyname#i>
351         #propertyValues[#i] = <propertyvalue#i>
352     }
353 &smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
354 &smEnd;
```

355 6.1.3.3 Set Base System Time

356 This section describes how to set the time on the base system. There are two possible property
 357 name/value pairs that will result in the setting of the base system time. When the `currenttime` property
 358 is specified, the format is expected to be in regular date-time format as defined in [DSP0216](#). When the
 359 `currenttime#time` property is specified, the format for the property value is defined in [DSP0216](#).

360 6.1.3.3.1 Command Form

```

361 set <CIM_ComputerSystem single object> ( currenttime=<mof time value> /
362     currenttime#time=<friendly time value>)
```

363 6.1.3.3.2 CIM Requirements

364 See `CIM_ComputerSystem` and `CIM_TimeService.ManageTime()` in the “CIM Elements” section of the
 365 [Computer System Profile](#) for the list of mandatory properties.

366 6.1.3.3.3 Behavior Requirements

367 6.1.3.3.3.1 Preconditions

368 `#requestedtime` contains the `datetime` which corresponds to the value of the `currenttime` or
 369 `currenttime#time` property value.

370 6.1.3.3.3.2 Pseudo Code

```

371 $instance=<CIM_ComputerSystem single object>
372 //Find the TimeService associated to the target system
373 #Error = smOpAssociators(
374     $instance.getObjectPath(),
375     "CIM_ServiceAffectsElement",
376     "CIM_TimeService",
377     NULL,
378     NULL,
379     $Services[]);
380 //if there isn't a service associated the function is not supported.
381 if (0 == $Services[.length]){
382     //unsupported
383     $OperationError = smNewInstance("CIM_Error");
384     //CIM_ERR_NOT_SUPPORTED
385     $OperationError.CIMStatusCode = 7;
386     //Other
```

```

387     $OperationError.ErrorType = 1;
388     //Low
389     $OperationError.PerceivedSeverity = 2;
390     $OperationError.OwningEntity = DMTF:SMCLP;
391     $OperationError.MessageID = 0x00000001;
392     $OperationError.Message = "Operation is not supported.";
393     &smAddError($job, $OperationError);
394     &smMakeCommandStatus($job);
395     &smEnd;
396 }
397 $Service-> = $Services[0].getObjectPath();
398 %InArguments[] = { newArgument("GetRequest", FALSE)
399                   newArgument("TimeData", #time),
400                   newArgument("ManagedElement", $instance.getObjectPath())}
401 %OutArguments[] = { };
402 //invoke method
403 #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
404     "ManageTime",
405     %InArguments[],
406     %OutArguments[]);
407 // process return code to CLP Command Status
408 if (0 != #Error.code) {
409     //method invocation failed
410     if ( (null != #Error.$error) && (null != #Error.$error[0]) )
411     {
412         // if the method invocation contains an embedded error
413         // use it for the Error for the overall job
414         &smAddError($job, #Error.$error[0]);
415         &smMakeCommandStatus($job);
416         &smEnd;
417     }
418     else if (#Error.code == 17)
419     {
420         //trap for CIM_METHOD_NOT_FOUND
421         //and make nice Unsupported msg.
422         //unsupported
423         $OperationError = smNewInstance("CIM_Error");
424         //CIM_ERR_NOT_SUPPORTED
425         $OperationError.CIMStatusCode = 7;
426         //Other
427         $OperationError.ErrorType = 1;
428         //Low
429         $OperationError.PerceivedSeverity = 2;
430         $OperationError.OwningEntity = DMTF:SMCLP;
431         $OperationError.MessageID = 0x00000001;
432         $OperationError.Message = "Operation is not supported.";
433         &smAddError($job, $OperationError);
434         &smMakeCommandStatus($job);
435         &smEnd;

```

```
436 }
437 Else
438 {
439 //operation failed, but no detailed error instance, need to make one up
440 //make an Error instance and associate with job for Operation
441 $OperationError = smNewInstance("CIM_Error");
442 //CIM_ERR_FAILED
443 $OperationError.CIMStatusCode = 1;
444 //Software Error
445 $OperationError.ErrorType = 4;
446 //Unknown
447 $OperationError.PerceivedSeverity = 0;
448 $OperationError.OwningEntity = DMTF:SMCLP;
449 $OperationError.MessageID = 0x00000009;
450 $OperationError.Message = "An internal software error has occurred.";
451 &smAddError($job, $OperationError);
452 &smMakeCommandStatus($job);
453 &smEnd;
454 }
455 }//if CIM op failed
456 else if (0 == #returnStatus){
457 //completed successfully
458 %InArguments[] = { newArgument("GetRequest", TRUE)
459 newArgument("ManagedElement", $instance.GetObjectPath())}
460 %OutArguments[] = { newArgument("TimeData", #timedata)};
461 //invoke method
462 #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
463 "ManageTime",
464 %InArguments[],
465 %OutArguments[]);
466 if (0 != #Error.code)
467 {
468 //method invocation failed
469 if ( (null != #Error.$error) && (null != #Error.$error[0]) )
470 {
471 // if the method invocation contains an embedded error
472 // use it for the Error for the overall job
473 &smAddError($job, #Error.$error[0]);
474 &smMakeCommandStatus($job);
475 &smEnd;
476 }
477 }
478 Else
479 {
480 //generic failure
481 $OperationError = smNewInstance("CIM_Error");
482 //CIM_ERR_FAILED
483 $OperationError.CIMStatusCode = 1;
484 //Other
```



```

485     $OperationError.ErrorType = 1;
486     //Low
487     $OperationError.PerceivedSeverity = 2;
488     $OperationError.OwningEntity = DMTF:SMCLP;
489     $OperationError.MessageID = 0x00000002;
490     $OperationError.Message = "Failed. No further information is available.";
491     &smAddError($job, $OperationError);
492     &smMakeCommandStatus($job);
493 }
494 &smEnd;
495 }
496 else if (1 == #returnStatus)
497 {
498     //unsupported
499     $OperationError = smNewInstance("CIM_Error");
500     //CIM_ERR_NOT_SUPPORTED
501     $OperationError.CIMStatusCode = 7;
502     //Other
503     $OperationError.ErrorType = 1;
504     //Low
505     $OperationError.PerceivedSeverity = 2;
506     $OperationError.OwningEntity = DMTF:SMCLP;
507     $OperationError.MessageID = 0x00000001;
508     $OperationError.Message = "Operation is not supported.";
509     &smAddError($job, $OperationError);
510     &smMakeCommandStatus($job);
511     &smEnd;
512 }
513 Else
514 {
515     //generic failure
516     $OperationError = smNewInstance("CIM_Error");
517     //CIM_ERR_FAILED
518     $OperationError.CIMStatusCode = 1;
519     //Other
520     $OperationError.ErrorType = 1;
521     //Low
522     $OperationError.PerceivedSeverity = 2;
523     $OperationError.OwningEntity = DMTF:SMCLP;
524     $OperationError.MessageID = 0x00000002;
525     $OperationError.Message = "Failed. No further information is available.";
526     &smAddError($job, $OperationError);
527     &smMakeCommandStatus($job);
528 }

```

529 6.1.4 Show

530 This section describes how to implement the `show` verb when applied to an instance of
531 `CIM_ComputerSystem`. Implementations shall support the use of the `show` verb with
532 `CIM_ComputerSystem`.

533 6.1.4.1 Show Command Form for Multiple Instances Target

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

535 6.1.4.1.1 Command Form

```
536 show <CIM_ComputerSystem multiple instances>
```

537 6.1.4.1.2 CIM Requirements

538 See CIM_ComputerSystem in the “CIM Elements” section of the [Computer System Profile](#) for the list of
539 mandatory properties.

540 6.1.4.1.3 Behavior Requirements

541 6.1.4.1.3.1 Preconditions

542 \$containerInstance represents the instance of a sub-class of CIM_System which represents the
543 container system and is associated to the targeted instances of CIM_ComputerSystem through the
544 CIM_SystemComponent association.

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

546 6.1.4.1.3.2 Pseudo Code

```
547 #propertylist[] = NULL;
548 if (false == #all)
549     {
550         #propertylist[] = {<all mandatory non-key properties>
551     }
552 // Step 1 - find all the scoped instances
553     #Error = &smOpAssociators (
554         $containerInstance.getObjectPath(),
555         "SystemComponent",
556         "CIM_ComputerSystem",
557         NULL,
558         NULL,
559         NULL,
560         $instances[] );
561 if (0 != #Error.code)
562     {
563         &smProcessOpError (#Error);
564         //includes &smEnd;
565     }
566 // Step 2 - add their referenced properties
567 for $instance in $instances[]
568     {
569         &lAddReferencedProperties ( $instance, #referencedPropertyName[] );
570     }
571 //step 3 - display them
572 &smShowInstancesWithReferencedProperties (
573     $instances[],
574     #propertyList[],
575     #referencedPropertyName[] );
576 &smEnd;
```

577 6.1.4.2 Show Command Form for a Single Instance Target

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

579 6.1.4.2.1 Command Form

```
580 show <CIM_ComputerSystem single instance>
```

581 6.1.4.2.2 CIM Requirements

582 See CIM_ComputerSystem in the “CIM Elements” section of the [Computer System Profile](#) for the list of
583 mandatory properties.

584 6.1.4.2.3 Behavior Requirements

585 6.1.4.2.3.1 Preconditions

586 \$instance represents the targeted instance of CIM_ComputerSystem.

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

588 6.1.4.2.3.2 Pseudo Code

```
589 $instance=<CIM_ComputerSystem single instance>;
590 #propertylist[] = NULL;
591 if (false == #all)
592 {
593     #propertylist[] = <array of mandatory non-key property names (see CIM
594     Requirements)>;
595 }
596 &lAddReferencedProperties ($instance, $referencedPropertyNames[], #all );
597 &smShowInstanceWithReferencedProperties ( $device, #propertyList[],
598     $referencedPropertyNames[] );
599 &smEnd;
```

600 6.1.5 Start

601 This section describes how to implement the `start` verb when applied to an instance of
602 CIM_ComputerSystem. Implementations may support the use of the `start` verb with
603 CIM_ComputerSystem.

604 6.1.5.1 Command Form

```
605 start <CIM_ComputerSystem single instance>
```

606 6.1.5.2 CIM Requirements

```
607 uint16 EnabledState;
608 uint16 RequestedState;
609 uint32 CIM_ComputerSystem.RequestStateChange (
610     [IN] uint16 RequestedState,
611     [OUT] REF CIM_ConcreteJob Job,
612     [IN] datetime TimeoutPeriod );
```

613 6.1.5.3 Behavior Requirements

614 6.1.5.3.1.1 Preconditions

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

616 6.1.5.3.1.2 Pseudo Code

```
617 $instance=<CIM_ComputerSystem single instance>;  
618 &smStartRSC ( $instance.getObjectPath() );  
619 &smEnd;
```

620 6.1.6 Stop

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

624 6.1.6.1 Command Form

```
625 stop <CIM_ComputerSystem single instance>
```

626 6.1.6.2 CIM Requirements

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

633 6.1.6.3 Behavior Requirements

634 6.1.6.3.1 Preconditions

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

636 6.1.6.3.2 Pseudo Code

```
637 $instance=<CIM_ComputerSystem single instance>;  
638 &smStopRSC ( $instance.getObjectPath() );  
639 &smEnd;
```

640 6.2 CIM_ElementCapabilities

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

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

647

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	

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

650 6.2.1 Ordering of Results

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

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

654 6.2.2 Show

655 This section describes how to implement the `show` verb when applied to an instance of
656 CIM_ElementCapabilities. Implementations shall support the use of the `show` verb with
657 CIM_ElementCapabilities.

658 6.2.2.1 Show Command Form for Multiple Instances Target – 659 CIM_EnabledLogicalElementCapabilities Reference

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

663 6.2.2.1.1 Command Form

```
664 show <CIM_ElementCapabilities multiple instances>
```

665 6.2.2.1.2 CIM Requirements

666 See CIM_ElementCapabilities in the “CIM Elements” section of the [Computer System Profile](#) for the list of
667 mandatory properties.

668 6.2.2.1.3 Behavior Requirements

669 6.2.2.1.3.1 Preconditions

670 `instance` represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by
671 CIM_ElementCapabilities.

672 6.2.2.1.3.2 Pseudo Code

```
673 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
674 &smEnd;
```

675 6.2.2.2 Show Command Form for a Single Instance – CIM_ComputerSystem Reference

676 This command form is used to show a single instance of CIM_ElementCapabilities. This command form
677 corresponds to a `show` command issued against a single instance of CIM_ElementCapabilities where
678 only one reference is specified and the reference is to the instance of CIM_ComputerSystem.

679 6.2.2.2.1 Command Form

```
680 show <CIM_ElementCapabilities single instance>
```

681 6.2.2.2.2 CIM Requirements

682 See CIM_ElementCapabilities in the “CIM Elements” section of the [Computer System Profile](#) for the list of
683 mandatory properties.

684 6.2.2.2.3 Behavior Requirements

685 6.2.2.2.3.1 Preconditions

686 `$instance` represents the instance of CIM_ComputerSystem which is referenced by
687 CIM_ElementCapabilities.

688 6.2.2.2.3.2 Pseudo Code

```
689 &smShowAssociationInstances ( "CIM_ElementCapabilities", $instance.getObjectPath() );  
690 &smEnd;
```

691 6.2.2.3 Show Command Form for a Single Instance Target – Both References

692 This command form is for the `show` verb applied to a single instance. This command form corresponds to
693 the `show` command issued against CIM_ElementCapabilities where both references are specified;
694 therefore, the desired instance is unambiguously identified.

695 6.2.2.3.1 Command Form

```
696 show <CIM_ElementCapabilities single instance>
```

697 6.2.2.3.2 CIM Requirements

698 See CIM_ElementCapabilities in the “CIM Elements” section of the [Computer System Profile](#) for the list of
699 mandatory properties.

700 6.2.2.3.3 Behavior Requirements

701 6.2.2.3.3.1 Preconditions

702 `$instanceA` represents the referenced instance of CIM_ComputerSystem through
703 CIM_ElementCapabilities association.

704 `$instanceB` represents the instance of CIM_EnabledLogicalElementCapabilities which is referenced by
705 CIM_ElementCapabilities.

706 **6.2.2.3.3.2 Pseudo Code**

```

707 &smShowAssociationInstance ( "CIM_ElementCapabilities", $instanceA.getObjectPath(),
708     $instanceB.getObjectPath() );
709 &smEnd;

```

710 **6.3 CIM_EnabledLogicalElementCapabilities**

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

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

717 **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	

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

720 **6.3.1 Ordering of Results**

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

- 723 • Results for `CIM_EnabledLogicalElementCapabilities` are unordered; therefore, no algorithm is
 724 defined.

725 **6.3.2 Show**

726 This section describes how to implement the `show` verb when applied to an instance of
 727 `CIM_EnabledLogicalElementCapabilities`. Implementations shall support the use of the `show` verb with
 728 `CIM_EnabledLogicalElementCapabilities`.

729 **6.3.2.1 Show Command Form for Multiple Instances Target**

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

731 6.3.2.1.1 Command Form

```
732 show <CIM_EnabledLogicalElementCapabilities multiple instances>
```

733 6.3.2.1.2 CIM Requirements

734 See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Computer System](#)
735 [Profile](#) for the list of mandatory properties.

736 6.3.2.1.3 Behavior Requirements

737 6.3.2.1.3.1 Preconditions

738 \$containerInstance represents the instance of CIM_ConcreteCollection with ElementName property
739 that contains “Capabilities” and is associated to the targeted instances of
740 CIM_EnabledLogicalElementCapabilities through the CIM_MemberOfCollection association.

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

742 6.3.2.1.3.2 Pseudo Code

```
743 #propertylist[] = NULL;  
744 if (false == #all)  
745     {  
746         #propertylist[] = <array of mandatory non-key property names (see CIM  
747             Requirements)>;  
748     }  
749 &smShowInstances ( "CIM_EnabledLogicalElementCapabilities", "CIM_MemberOfCollection",  
750     $containerInstance.getObjectPath(), #propertylist[] );  
751 &smEnd;
```

752 6.3.2.2 Show Command Form for a Single Instance Target

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

754 6.3.2.2.1 Command Form

```
755 show <CIM_EnabledLogicalElementCapabilities single instance>
```

756 6.3.2.2.2 CIM Requirements

757 See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the [Computer System](#)
758 [Profile](#) for the list of mandatory properties.

759 6.3.2.2.3 Behavior Requirements

760 6.3.2.2.3.1 Preconditions

761 \$instance represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

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

763 **6.3.2.2.3.2 Pseudo Code**

```

764 $instance=<CIM_EnabledLogicalElementCapabilities single instance>;
765 #propertylist[] = NULL;
766 if (false == #all)
767 {
768     #propertylist[] = <array of mandatory non-key property names (see CIM
769     Requirements)>;
770 }
771 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );
772 &smEnd;
    
```

773 **6.4 CIM_HostedService**

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

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

780 **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	

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

783 **6.4.1 Ordering of Results**

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

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

787 **6.4.2 Show**

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

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

791 6.4.2.1 Show Multiple Instances

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

795 6.4.2.1.1 Command Form

```
796 show <CIM_HostedService multiple instances>
```

797 6.4.2.1.2 CIM Requirements

798 See `CIM_HostedService` in the “CIM Elements” section of the [Computer System Profile](#) for the list of
799 mandatory properties.

800 6.4.2.1.3 Behavior Requirements

801 6.4.2.1.4 Preconditions

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

803 6.4.2.1.5 Pseudo Code

```
804 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );  
805 &smEnd;
```

806 6.4.2.2 Show a Single Instance – CIM_TimeService Reference

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

811 6.4.2.2.1 Command Form

```
812 show <CIM_HostedService single instance>
```

813 6.4.2.2.2 CIM Requirements

814 See `CIM_HostedService` in the “CIM Elements” section of the [Computer System Profile](#) for the list of
815 mandatory properties.

816 6.4.2.2.3 Behavior Requirements

817 6.4.2.2.3.1 Preconditions

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

819 6.4.2.2.3.2 Pseudo Code

```
820 &smShowAssociationInstances ( "CIM_HostedService", $instance.getObjectPath() );  
821 &smEnd;
```

822 6.4.2.3 Show a Single Instance – Both References

823 This command form is for the `show` verb applied to a single instance. This command form corresponds to
824 the `show` command issued against `CIM_HostedService` where both references are specified; therefore,
825 the desired instance is unambiguously identified.

826 **6.4.2.3.1 Command Form**

827 `show <CIM_HostedService single instance>`

828 **6.4.2.3.2 CIM Requirements**

829 See CIM_HostedService in the “CIM Elements” section of the [Computer System Profile](#) for the list of
830 mandatory properties.

831 **6.4.2.3.3 Behavior Requirements**

832 **6.4.2.3.3.1 Preconditions**

833 \$instanceA contains the instance of CIM_ComputerSystem which is referenced by
834 CIM_HostedService.

835 \$instanceB contains the instance of CIM_TimeService which is referenced by CIM_HostedService.

836 **6.4.2.3.3.2 Pseudo Code**

```
837 &smShowAssociationInstance ( "CIM_HostedService", $instanceA.getObjectPath(),
838     $instanceB.getObjectPath() );
839 &smEnd;
```

840 **6.5 CIM_ServiceAffectsElement**

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

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

847 **Table 6 – 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 6.5.2.
start	Not supported	
stop	Not supported	

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

850 6.5.1 Ordering of Results

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

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

854 6.5.2 Show

855 This section describes how to implement the `show` verb when applied to an instance of
856 CIM_ServiceAffectsElement. Implementations shall support the use of the `show` verb with
857 CIM_ServiceAffectsElement.

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

860 6.5.2.1 Show a Single Instance – CIM_TimeService Reference

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

864 6.5.2.1.1 Command Form

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

866 6.5.2.1.2 CIM Requirements

867 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Computer System Profile](#) for the list
868 of mandatory properties.

869 6.5.2.1.3 Behavior Requirements

870 6.5.2.1.3.1 Preconditions

871 `$instance` contains the instance of CIM_TimeService which is referenced by
872 CIM_ServiceAffectsElement.

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

874 6.5.2.1.3.2 Pseudo Code

```
875 #propertylist[] = NULL;  
876 if (false == #all)  
877 {  
878     #propertylist[] = {/all mandatory non-key properties};  
879 }  
880 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),  
881     #propertylist[] );  
882 &smEnd;
```

883 6.5.2.2 Show a Single Instance – CIM_ComputerSystem Reference

884 This command form is for the `show` verb applied to a single instance. This command form corresponds to
885 the `show` command issued against CIM_ServiceAffectsElement where the reference specified is to an
886 instance of CIM_ComputerSystem. An instance of CIM_TimeService is associated with exactly one
887 CIM_ComputerSystem instance; thus, a single instance will be returned.

888 6.5.2.2.1 Command Form

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

890 6.5.2.2.2 CIM Requirements

891 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Computer System Profile](#) for the list
892 of mandatory properties.

893 6.5.2.2.3 Behavior Requirements

894 6.5.2.2.3.1 Preconditions

895 \$instance contains the instance of CIM_ComputerSystem which is referenced by
896 CIM_ServiceAffectsElement.

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

898 6.5.2.2.3.2 Pseudo Code

```
899 #propertylist[] = NULL;
900 if (false == #all)
901     {
902         #propertylist[] = {/all mandatory non-key properties};
903     }
904 &smShowAssociationInstances ( "CIM_ServiceAffectsElement", $instance.getObjectPath(),
905     #propertylist[] );
906 &smEnd;
```

907 6.5.2.3 Show a Single Instance – Both References

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

911 6.5.2.3.1 Command Form

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

913 6.5.2.3.2 CIM Requirements

914 See CIM_ServiceAffectsElement in the “CIM Elements” section of the [Computer System Profile](#) for the list
915 of mandatory properties.

916 6.5.2.3.3 Behavior Requirements

917 6.5.2.3.3.1 Preconditions

918 \$instanceA contains the instance of CIM_ServiceAvailableToElement which is referenced by
919 CIM_ServiceAffectsElement.

920 \$instanceB contains the instance of CIM_ComputerSystem which is referenced by
921 CIM_ServiceAffectsElement.

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

923 **6.5.2.3.3.2 Pseudo Code**

```

924 #propertylist[] = NULL;
925 if (false == #all)
926     {
927         #propertylist[] = { //all mandatory non-key properties};
928     }
929 &smShowAssociationInstance ( "CIM_ServiceAffectsElement", $instanceA.getObjectPath(),
930     $instanceB.getObjectPath(), #propertylist[] );
931 &smEnd;

```

932 **6.6 CIM_TimeService**

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

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

939 **Table 7 – 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 6.6.2.
start	Not supported	
stop	Not supported	

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

942 **6.6.1 Ordering of Results**

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

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

946 **6.6.2 Show**

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

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

950 6.6.2.1 Show a Single Instance

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

952 6.6.2.1.1 Command Form

```
953 show <CIM_TimeService single instance>
```

954 6.6.2.1.2 CIM Requirements

955 See `CIM_TimeService` in the “CIM Elements” section of the [Computer System Profile](#) for the list of
956 mandatory properties.

957 6.6.2.1.3 Behavior Requirements

958 6.6.2.1.3.1 Preconditions

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

960 6.6.2.1.3.2 Pseudo Code

```
961 $instance=<CIM_TimeService single instance>  
962 #propertylist[] = NULL;  
963 if (false == #all)  
964     {  
965         #propertylist[] = {/all mandatory non-key properties};  
966     }  
967 &smShowInstance ( $instance.getObjectPath(), #propertylist[] );  
968 &smEnd;
```

969

970
971
972
973
974

ANNEX A

(informative)

Change Log

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

975