Service Processor Profile SM CLP Command Mapping Specification

5
6

10

33

CONTENTS

Tables

71

72   Foreword

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

Conventions

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

Acknowledgements

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

81 • Aaron Merkin – IBM

82 • Jon Hass – Dell

83 • Khachatur Papanyan – Dell

84 • Jeff Hilland – HP

85 • Christina Shaw – HP

86 • Perry Vincent – Intel

87 • John Leung – Intel

88

89   Introduction

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

94   The target audience for this specification is implementers of the SM CLP support for the *Service*
95   *Processor Profile*.

Service Processor Profile SM CLP Command Mapping Specification

1 Scope

This specification contains the requirements for an implementation of the SM CLP to provide access to, and implement the behaviors of, the *Service Processor Profile*.

2 Normative References

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

2.1 Approved References

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

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

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

2.1 Other References

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

3 Terms and Definitions

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

3.1
can
used for statements of possibility and capability, whether material, physical, or causal

3.2
cannot
used for statements of possibility and capability, whether material, physical or causal

3.3
conditional
indicates requirements to be followed strictly in order to conform to the document when the specified conditions are met

127 **3.4**
128 **mandatory**
129 indicates requirements to be followed strictly in order to conform to the document and from which no
130 deviation is permitted

131 **3.5**
132 **may**
133 indicates a course of action permissible within the limits of the document

134 **3.6**
135 **need not**
136 indicates a course of action permissible within the limits of the document

137 **3.7**
138 **optional**
139 indicates a course of action permissible within the limits of the document
140

141 **3.8**
142 **shall**
143 indicates requirements to be followed strictly in order to conform to the document and from which no
144 deviation is permitted

145 **3.9**
146 **shall not**
147 indicates requirements to be followed strictly in order to conform to the document and from which no
148 deviation is permitted

149 **3.10**
150 **should**
151 indicates that among several possibilities, one is recommended as particularly suitable, without
152 mentioning or excluding others, or that a certain course of action is preferred but not necessarily required

153 **3.11**
154 **should not**
155 indicates that a certain possibility or course of action is deprecated but not prohibited

156 # 4   Symbols and Abbreviated Terms

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

158 **4.1**
159 **CIM**
160 Common Information Model

161 **4.2**
162 **CLP**
163 Command Line Protocol

164 **4.3**
165 **DMTF**
166 Distributed Management Task Force

167   **4.4**

168   **SM**

169   Server Management

170   **4.5**

171   **SMI-S**

172   Storage Management Initiative Specification

173   **4.6**

174   **SNIA**

175   Storage Networking Industry Association

176   **4.7**

177   **UFsT**

178   User Friendly Selection Tag

179   5   Recipes

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

182   •   smResetRSC

183   •   smShowInstance

184   •   smShowInstances

185   •   smShowAssociationInstance

186   •   smShowAssociationInstances

187   •   smStartRSC

188   •   smStopRSC

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

191   **Table 1 – Local Recipes**

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

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

193   5.1   IAddReferencedProperties

194   5.1.1   Description

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

196   5.1.2   Preconditions

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

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

5.1.3 Pseudo Code

```
199
200  sub lAddReferencedProperties($instance, #ReferencedPropertyNames[]){
201      #propertylist[] = NULL;
202  //find the associated service, call the method as a query, insert the time as a
203      referenced property
204  // Find the TimeService associated to the target system
205  #Error = smOpAssociators(
206      $instance.getObjectPath(),
207      "CIM_ServiceAffectsElement",
208      "CIM_TimeService",
209      NULL,
210      NULL,
211      $Services[]);
212  //if there is not a service associated the function is not supported.
213  if (0 == $Services[].length){
214      //not supported, don't add property
215          return;
216  }
217  $Service-> = $Services[0].getObjectPath();
218   %InArguments[] = { newArgument("GetRequest", FALSE)
219                      newArgument("ManagedElement", $instance.getOjectPath())}
220  %OutArguments[] = { newArgument("TimeData", #timedata)};
221  //invoke method
222  #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
223      "ManageTime",
224      %InArguments[],
225      %OutArguments[]);
226
227  // process return code to CLP Command Status
228  if (0 != #Error.code)   {
229      //method invocation failed
230      if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
231          // if the method invocation contains an embedded error
232          // use it for the Error for the overall job
233          &smAddError($job, #Error.$error[0]);
234          &smMakeCommandStatus($job);
235          &smEnd;
236      }
237      else if (#Error.code == 17)   {
238          //not supported, so don't add property
239          return;
240      }
241      else {
242          //operation failed, but no detailed error instance, need to make one up
243          //make an Error instance and associate with job for Operation
244              $OperationError = smNewInstance("CIM_Error");
245              //CIM_ERR_FAILED
246              $OperationError.CIMStatusCode = 1;
247              //Software Error
```

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

284 6   Mappings

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

288 6.1   CIM_ComputerSystem

289 The cd, exit, help, and version verbs shall be supported as described in DSP0216.

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

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

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

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

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

297 6.1.1 Ordering of Results

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

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

301 6.1.2 Reset

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

305 6.1.2.1 Command Form

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

#### 307  6.1.2.2 CIM Requirements

308  ```
uint16 EnabledState;
```
309 ```
uint16 RequestedState;
```
310  ```
uint32 CIM_ComputerSystem.RequestStateChange (
```
311 ```
    [IN] uint16 RequestedState,
```
312  ```
    [OUT] REF CIM_ConcreteJob Job,
```
313 ```
    [IN] datetime TimeoutPeriod);
```

#### 314  6.1.2.3 Behavior Requirements

#### 315  6.1.2.3.1 Preconditions

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

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

318 **6.1.2.3.2 Pseudo Code**

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

321 ## 6.1.3   Set

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

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

326 **6.1.3.1 General Usage of Set for a Single Property**

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

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

332 **6.1.3.1.1   Command Form**

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

334 **6.1.3.1.2   CIM Requirements**

335 See CIM_ComputerSystem in the "CIM Elements" section of the *Service Processor Profile* for the list of
336 mandatory properties.

337 **6.1.3.1.3   Behavior Requirements**

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

343 **6.1.3.2 General Usage of Set for Multiple Properties**

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

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

350 **6.1.3.2.1   Command Form**

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

353 **6.1.3.2.2   CIM Requirements**

354 See CIM_ComputerSystem in the "CIM Elements" section of the *Service Processor Profile* for the list of
355 mandatory properties.

356 **6.1.3.2.3 Behavior Requirements**

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

366 **6.1.3.3 Set RequestedState to "Offline"**

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

369 **6.1.3.3.1 Command Form**

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

371 **6.1.3.3.2 CIM Requirements**

372 See CIM_ComputerSystem in the "CIM Elements" section of the *Service Processor Profile* for the list of
373 mandatory properties.

374 **6.1.3.3.3 Behavior Requirements**

375 **6.1.3.3.3.1 Preconditions**

376 $instance represents the targeted instance of CIM_ComputerSystem.

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

378 **6.1.3.3.3.2 Pseudo Code**

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

382 **6.1.3.4 Set Service Processor Time**

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

388 **6.1.3.4.1 Command Form**

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

391 **6.1.3.4.2 CIM Requirements**

392 See CIM_ComputerSystem and CIM_TimeService.ManageTime( ) in the "CIM Elements" section of the
393 *Service Processor Profile* for the list of mandatory properties.

394 6.1.3.4.3 Behavior Requirements

395 6.1.3.4.3.1 Preconditions

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

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

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

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

6.1.4   Show

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

6.1.4.1   Show Command Form for Multiple Instances Target

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

6.1.4.1.1   Command Form

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

6.1.4.1.2   CIM Requirements

See CIM_ComputerSystem in the "CIM Elements" section of the *Service Processor Profile* for the list of
mandatory properties.

6.1.4.1.3   Behavior Requirements

6.1.4.1.3.1   Preconditions

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

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

6.1.4.1.3.2   Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all )
    {
    #propertylist[] = { <all mandatory non-key properties}
    }

// Step 1 - find all the scoped instances
    #Error = &smOpAssociators (
        $containerInstance.getObjectPath(),
        "SystemComponent",
        "CIM_ComputerSystem",
        NULL,
        NULL,
        NULL,
        $instances[] );
```

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

597 6.1.4.2   Show Command Form for a Single Instance Target

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

599 6.1.4.2.1   Command Form

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

601 6.1.4.2.2   CIM Requirements

602 See CIM_ComputerSystem in the "CIM Elements" section of the *Service Processor Profile* for the list of
603 mandatory properties.

604 6.1.4.2.3   Behavior Requirements

605 6.1.4.2.3.1   Preconditions

606 $instance represents the targeted instance of CIM_ComputerSystem.

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

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

609 6.1.4.2.3.2   Pseudo Code

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

619 **6.1.5 Start**

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

623 **6.1.5.1 Command Form**

624 **start <CIM_ComputerSystem *single instance*>**

625 **6.1.5.2 CIM Requirements**

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

632 **6.1.5.3 Behavior Requirements**

633 **6.1.5.3.1 Preconditions**

634 $instance represents the targeted instance of CIM_ComputerSystem.

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

636 **6.1.5.3.1.1 Pseudo Code**

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

639 **6.1.6 Stop**

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

643 **6.1.6.1 Command Form**

644 **stop <CIM_ComputerSystem *single instance*>**

645 **6.1.6.2 CIM Requirements**

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

652 **6.1.6.3 Behavior Requirements**

653 **6.1.6.3.1.1 Preconditions**

654 $instance represents the targeted instance of CIM_ComputerSystem.

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

656   **6.1.6.3.1.2   Pseudo Code**

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

658   `&smEnd;`

659   ## 6.2   CIM_ElementCapabilities

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

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

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

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

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

669   **6.2.1   Ordering of Results**

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

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

673   **6.2.2   Show**

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

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

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

682   **6.2.2.1.1   Command Form**

683   `show <CIM_ElementCapabilities multiple instances>`

684 **6.2.2.1.2 CIM Requirements**

685 See CIM_ElementCapabilities in the "CIM Elements" section of the *Service Processor Profile* for the list of
686 mandatory properties.

687 **6.2.2.1.3 Behavior Requirements**

688 **6.2.2.1.3.1 Preconditions**

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

691 **6.2.2.1.3.2 Pseudo Code**

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

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

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

698 **6.2.2.2.1 Command Form**

699 `show <CIM_ElementCapabilities single instance>`

700 **6.2.2.2.2 CIM Requirements**

701 See CIM_ElementCapabilities in the "CIM Elements" section of the *Service Processor Profile* for the list of
702 mandatory properties.

703 **6.2.2.2.3 Behavior Requirements**

704 **6.2.2.2.3.1 Preconditions**

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

707 **6.2.2.2.3.2 Pseudo Code**

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

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

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

714 **6.2.2.3.1 Command Form**

715 `show <CIM_ElementCapabilities single instance>`

716 **6.2.2.3.2 CIM Requirements**

717 See CIM_ElementCapabilities in the "CIM Elements" section of the *Service Processor Profile* for the list of
718 mandatory properties.

719 **6.2.2.3.3   Behavior Requirements**

720 **6.2.2.3.3.1   Preconditions**

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

724 **6.2.2.3.3.2   Pseudo Code**

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

728   6.3   CIM_EnabledLogicalElementCapabilities

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

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

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

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

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

738   6.3.1   Ordering of Results

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

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

743 **6.3.2 Show**

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

747 **6.3.2.1 Show Command Form for Multiple Instances Target**

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

749 **6.3.2.1.1 Command Form**

750 `show <CIM_EnabledLogicalElementCapabilities multiple instances>`

751 **6.3.2.1.2 CIM Requirements**

752 See CIM_EnabledLogicalElementCapabilities in the "CIM Elements" section of the *Service Processor*
753 *Profile* for the list of mandatory properties.

754 **6.3.2.1.3 Behavior Requirements**

755 **6.3.2.1.3.1 Preconditions**

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

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

760 **6.3.2.1.3.2 Pseudo Code**

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

770 **6.3.2.2 Show Command Form for a Single Instance Target**

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

772 **6.3.2.2.1 Command Form**

773 `show <CIM_EnabledLogicalElementCapabilities single instance>`

774 **6.3.2.2.2 CIM Requirements**

775 See CIM_EnabledLogicalElementCapabilities in the "CIM Elements" section of the *Service Processor*
776 *Profile* for the list of mandatory properties.

777   **6.3.2.2.3   Behavior Requirements**

778   **6.3.2.2.3.1   Preconditions**

779   `$instance` represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

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

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

782   **6.3.2.2.3.2   Pseudo Code**

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

791   **6.4   CIM_HostedService**

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

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

798   **Table 5 – Command Verb Requirements for CIM_HostedService**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.4.2.
start	Not supported	
stop	Not supported	

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

801   **6.4.1   Ordering of Results**

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

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

805 **6.4.2 Show**

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

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

809 **6.4.2.1 Show Multiple Instances**

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

813 **6.4.2.1.1 Command Form**

814 `show <CIM_HostedService multiple instances>`

815 **6.4.2.1.2 CIM Requirements**

816 See CIM_HostedService in the "CIM Elements" section of the *Service Processor Profile* for the list of
817 mandatory properties.

818 **6.4.2.1.3 Behavior Requirements**

819 **6.4.2.1.3.1 Preconditions**

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

821 **6.4.2.1.3.2 Pseudo Code**

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

824 **6.4.2.2 Show a Single Instance – CIM_TimeService Reference**

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

829 **6.4.2.2.1 Command Form**

830 `show <CIM_HostedService single instance>`

831 **6.4.2.2.2 CIM Requirements**

832 See CIM_HostedService in the "CIM Elements" section of the *Service Processor Profile* for the list of
833 mandatory properties.

834 **6.4.2.2.3 Behavior Requirements**

835 **6.4.2.2.3.1 Preconditions**

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

837 **6.4.2.2.3.2 Pseudo Code**

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

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

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

844 **6.4.2.3.1 Command Form**

845 ```
show <CIM_HostedService single instance>
```

846 **6.4.2.3.2 CIM Requirements**

847 See CIM_HostedService in the "CIM Elements" section of the *Service Processor Profile* for the list of
848 mandatory properties.

849 **6.4.2.3.3 Behavior Requirements**

850 **6.4.2.3.3.1 Preconditions**

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

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

854 **6.4.2.3.3.2 Pseudo Code**

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

858 ## 6.5   CIM_IsSpare

859 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

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

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

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

868 ### 6.5.1   Ordering of Results

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

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

872 ### 6.5.2   Show

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

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

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

879 #### 6.5.2.1.1   Command Form

880 `show <CIM_IsSpare multiple instances>`

881 #### 6.5.2.1.2   CIM Requirements

882 See CIM_IsSpare in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory
883 properties.

884 #### 6.5.2.1.3   Behavior Requirements

885 #### 6.5.2.1.3.1   Preconditions

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

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

888 #### 6.5.2.1.3.2   Pseudo Code

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

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

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

902 #### 6.5.2.2.1   Command Form

903 `show <CIM_IsSpare single instance>`

904 **6.5.2.2.2 CIM Requirements**

905 See CIM_IsSpare in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory
906 properties.

907 **6.5.2.2.3 Behavior Requirements**

908 **6.5.2.2.3.1 Preconditions**

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

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

911 **6.5.2.2.3.2 Pseudo Code**

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

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

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

925 **6.5.2.3.1 Command Form**

926 `show <CIM_IsSpare single instance>`

927 **6.5.2.3.2 CIM Requirements**

928 See CIM_IsSpare in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory
929 properties.

930 **6.5.2.3.3 Behavior Requirements**

931 **6.5.2.3.3.1 Preconditions**

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

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

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

936 **6.5.2.3.3.2 Pseudo Code**

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

6.6   CIM_MemberOfCollection

947   The cd, exit, help, and version verbs shall be supported as described in DSP0216.

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

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

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

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

6.6.1   Ordering of Results

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

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

6.6.2   Show

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

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

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

968   **6.6.2.1.1   Command Form**

969   `show <CIM_MemberOfCollection multiple instances>`

970   **6.6.2.1.2   CIM Requirements**

971   See CIM_MemberOfCollection in the "CIM Elements" section of the *Service Processor Profile* for the list
972   of mandatory properties.

973   **6.6.2.1.3   Behavior Requirements**

974   **6.6.2.1.3.1   Preconditions**

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

977   **6.6.2.1.3.2   Pseudo Code**

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

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

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

984   **6.6.2.2.1   Command Form**

985   `show <CIM_MemberOfCollection single instance>`

986   **6.6.2.2.2   CIM Requirements**

987   See CIM_MemberOfCollection in the "CIM Elements" section of the *Service Processor Profile* for the list
988   of mandatory properties.

989   **6.6.2.2.3   Behavior Requirements**

990   **6.6.2.2.3.1   Preconditions**

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

993   **6.6.2.2.3.2   Pseudo Code**

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

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

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

1000 **6.6.2.3.1 Command Form**

1001 `show <CIM_MemberOfCollection` *`single instance`*`>`

1002 **6.6.2.3.2 CIM Requirements**

1003 See CIM_MemberOfCollection in the "CIM Elements" section of the *Service Processor Profile* for the list
1004 of mandatory properties.

1005 **6.6.2.3.3 Behavior Requirements**

1006 **6.6.2.3.3.1 Preconditions**

1007 `$instanceA` represents the referenced instance of CIM_ComputerSystem through
1008 CIM_MemberOfCollection association.

1009 `$instanceB` represents the instance of CIM_RedundancySet which is referenced by
1010 CIM_MemberOfCollection.

1011 **6.6.2.3.3.2 Pseudo Code**

1012 `&smShowAssociationInstance ( "CIM_MemberOfCollection", $instanceA.getObjectPath(),`
1013 `    $instanceB.getObjectPath() );`
1014 `&smEnd;`

6.7 CIM_OwningCollectionElement

1015

1016 The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

1017 Table 8 lists each SM CLP verb, the required level of support for the verb in conjunction with the target
1018 class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and
1019 target. Table 8 is for informational purposes only; in case of a conflict between Table 8 and requirements
1020 detailed in the following sections, the text detailed in the following sections supersedes the information in
1021 Table 8.

1022 **Table 8 – Command Verb Requirements for CIM_OwningCollectionElement**

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

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

6.7.1 Ordering of Results

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

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

6.7.2 Show

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

6.7.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

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

6.7.2.1.1 Command Form

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

6.7.2.1.2 CIM Requirements

See CIM_OwningCollectionElement in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory properties.

6.7.2.1.3 Behavior Requirements

6.7.2.1.3.1 Preconditions

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

6.7.2.1.3.2 Pseudo Code

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

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

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

6.7.2.2.1 Command Form

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

6.7.2.2.2 CIM Requirements

See CIM_OwningCollectionElement in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory properties.

1061   **6.7.2.2.3   Behavior Requirements**

1062   **6.7.2.2.3.1   Preconditions**

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

1065   **6.7.2.2.3.2   Pseudo Code**

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

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

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

1073   **6.7.2.3.1   Command Form**

1074   **show <CIM_OwningCollectionElement *single instance*>**

1075   **6.7.2.3.2   CIM Requirements**

1076   See CIM_OwningCollectionElement in the "CIM Elements" section of the *Service Processor Profile* for the
1077   list of mandatory properties.

1078   **6.7.2.3.3   Behavior Requirements**

1079   **6.7.2.3.3.1   Preconditions**

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

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

1084   **6.7.2.3.3.2   Pseudo Code**

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

1088   ## 6.8   CIM_RedundancySet

1089   The `cd`, `exit`, `help`, and `version` verbs shall be supported as described in DSP0216.

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

1095   **Table 9 – Command Verb Requirements for CIM_RedundancySet**

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

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

6.8.1   Ordering of Results

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

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

6.8.2   Set

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

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

6.8.2.1   Set Command for Failovers

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

6.8.2.1.1   Command Form

```
set <CIM_RedundancySet single instance> failoverfrom=<CIM_ComputerSystem single
    instance> failoverto=<CIM_ComputerSystem single instance>
```

6.8.2.1.2   CIM Requirements

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

6.8.2.1.3   Behavior Requirements

6.8.2.1.3.1   Preconditions

```
$instance=<CIM_RedundancySet single instance>
$FailoverFrom=<failoverfrom requested instance of CIM_ComputerSystem>
$FailoverTo=<failoverto requested instance of CIM_ComputerSystem>
```

6.8.2.1.3.2 Pseudo Code

```
1120
1121   %InArguments[] = { newArgument ( "FailoverFrom", $FailoverFrom.getObjectPath() ),
1122          newArgument ( "FailoverTo", $FailoverTo.getObjectPath () };
1123   %OutArguments[] = {};
1124   #Error = InvokeMethod ($target->,
1125                          "Failover",
1126                          %InArguments[],
1127                          %OutArguments[],
1128                          #returnStatus);
1129   if (0 != #Error.code)
1130   {
1131      //method invocation failed
1132      if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )
1133      {
1134         //if the method invocation contains an embedded error
1135         //use it for the Error for the overall job
1136         &smAddError($job, #Error.$error[0]);
1137         &smMakeCommandStatus($job);
1138         &smEnd;
1139      }
1140      else if ( 17 == #Error.code ) {
1141         //17 – CIM_ERR_METHOD_NOT_FOUND
1142         // The specified extrinsic method does not exist.
1143         $OperationError = smNewInstance("CIM_Error");
1144         // CIM_ERR_METHOD_NOT_FOUND
1145         $OperationError.CIMStatusCode = 17;
1146         //Software Error
1147         $OperationError.ErrorType = 10;
1148         //Unknown
1149         $OperationError.PerceivedSeverity = 0;
1150         $OperationError.OwningEntity = DMTF:SMCLP;
1151         $OperationError.MessageID = 0x00000001;
1152         $OperationError.Message = "Operation is not supported."
1153         &smAddError($job, $OperationError);
1154         &smMakeCommandStatus($job);
1155         &smEnd;
1156      }
1157      else
1158      {
1159         //operation failed, but no detailed error instance, need to make one up
1160         //make an Error instance and associate with job for Operation
1161         $OperationError = smNewInstance("CIM_Error");
1162         //CIM_ERR_FAILED
1163         $OperationError.CIMStatusCode = 1;
1164         //Software Error
1165         $OperationError.ErrorType = 4;
1166         //Unknown
1167         $OperationError.PerceivedSeverity = 0;
1168         $OperationError.OwningEntity = DMTF:SMCLP;
```

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

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

1228 6.8.2.2   General Usage of Set for a Single Property

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

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

1234 6.8.2.2.1   Command Form

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

1236 6.8.2.2.2   CIM Requirements

1237 See CIM_RedundancySet in the "CIM Elements" section of the *Service Processor Profile* for the list of
1238 mandatory properties.

1239 6.8.2.2.3   Behavior Requirements

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

1245 6.8.2.3   General Usage of Set for Multiple Properties

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

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

1252 6.8.2.3.1   Command Form

```
1253  set <CIM_RedundancySet single instance> <propertyname1>=<propertyvalue1>
1254      <propertynamen>=<propertyvaluen>
```

1255 6.8.2.3.2   CIM Requirements

1256 See CIM_RedundancySet in the "CIM Elements" section of the *Service Processor Profile* for the list of
1257 mandatory properties.

| 1258 | **6.8.2.3.3   Behavior Requirements** |

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

| 1268 | **6.8.3   Show** |

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

| 1271 | **6.8.3.1   Show Command Form for Multiple Instances Target** |

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

| 1273 | **6.8.3.1.1   Command Form** |

```
1274    show <CIM_RedundancySet multiple instances>
```

| 1275 | **6.8.3.1.2   CIM Requirements** |

1276 See CIM_RedundancySet in the "CIM Elements" section of the *Service Processor Profile* for the list of
1277 mandatory properties.

| 1278 | **6.8.3.1.3   Behavior Requirements** |

| 1279 | **6.8.3.1.3.1   Preconditions** |

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

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

| 1284 | **6.8.3.1.3.2   Pseudo Code** |

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

| 1294 | **6.8.3.2   Show Command Form for a Single Instance Target** |

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

1296 **6.8.3.2.1   Command Form**

1297 `show <CIM_RedundancySet single instance>`

1298 **6.8.3.2.2   CIM Requirements**

1299 See CIM_RedundancySet in the "CIM Elements" section of the *Service Processor Profile* for the list of
1300 mandatory properties.

1301 **6.8.3.2.3   Behavior Requirements**

1302 **6.8.3.2.3.1   Preconditions**

1303 In this section `$instance` represents the targeted instance of CIM_RedundancySet.

1304 `$instance=<CIM_RedundancySet single instance>;`

1305 `#all` is true if the "-all" option was specified with the command; otherwise, `#all` is false.

1306 **6.8.3.2.3.2   Pseudo Code**

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

1315 ## 6.9   CIM_ServiceAffectsElement

1316 The `cd` and `help` verbs shall be supported as described in DSP0216.

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

1322 **Table 10 – Command Verb Requirements for CIM_ServiceAffectsElement**

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.9.2.
start	Not supported	
stop	Not supported	

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

6.9.1   Ordering of Results

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

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

6.9.2   Show

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

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

6.9.2.1   Show a Single Instance – CIM_TimeService Reference

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

6.9.2.1.1   Command Form

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

6.9.2.1.2   CIM Requirements

See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory properties.

6.9.2.1.3   Behavior Requirements

6.9.2.1.3.1   Preconditions

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

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

6.9.2.1.3.2   Pseudo Code

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

6.9.2.2   Show Multiple Instances – CIM_ComputerSystem Reference

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

6.9.2.2.1   Command Form

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

6.9.2.2.2   CIM Requirements

See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory properties.

6.9.2.2.3   Behavior Requirements

6.9.2.2.3.1   Preconditions

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

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

6.9.2.2.3.2   Pseudo Code

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

6.9.2.3   Show a Single Instance – Both References

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

6.9.2.3.1   Command Form

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

6.9.2.3.2   CIM Requirements

See CIM_ServiceAffectsElement in the "CIM Elements" section of the *Service Processor Profile* for the list of mandatory properties.

6.9.2.3.3   Behavior Requirements

6.9.2.3.3.1   Preconditions

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

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

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

1397   **6.9.2.3.3.2   Pseudo Code**

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

1406   ## 6.10 CIM_TimeService

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

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

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

Command Verb	Requirement	Comments
create	Not supported	
delete	Not supported	
dump	Not supported	
load	Not supported	
reset	Not supported	
set	Not supported	
show	Shall	See 6.10.2.
start	Not supported	
stop	Not supported	

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

1416   ### 6.10.1 Ordering of Results

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

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

1420   ### 6.10.2 Show

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

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

1424   **6.10.2.1 Show a Single Instance**

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

1426   **6.10.2.1.1 Command Form**

1427   `show <CIM_TimeService single instance>`

1428   **6.10.2.1.2 CIM Requirements**

1429   See CIM_TimeService in the "CIM Elements" section of the *Service Processor Profile* for the list of
1430   mandatory properties.

1431   **6.10.2.1.3 Behavior Requirements**

1432   **6.10.2.1.3.1 Preconditions**

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

1434   **6.10.2.1.3.2 Pseudo Code**

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

1443   **ANNEX A**
1444   **(informative)**
1445
1446
1447   **Change Log**

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

1448