Document Number: DSP0824

Date: 2009-06-04

Version: 1.0.0

Service Processor Profile SM CLP Command Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

71

Foreword

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

Conventions

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

Acknowledgements

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

- Aaron Merkin – IBM
- Jon Hass – Dell
- Khachatur Papanyan – Dell
- Jeff Hilland – HP
- Christina Shaw – HP
- Perry Vincent – Intel
- John Leung – Intel

Introduction

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

The target audience for this specification is implementers of the SM CLP support for the *Service 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

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

3.5
may
indicates a course of action permissible within the limits of the document

3.6
need not
indicates a course of action permissible within the limits of the document

3.7
optional
indicates a course of action permissible within the limits of the document

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

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

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

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

4 Symbols and Abbreviated Terms

The following symbols and abbreviations are used in this document.

4.1
CIM
Common Information Model

4.2
CLP
Command Line Protocol

4.3
DMTF
Distributed Management Task Force

4.4
SM
Server Management

4.5
SMI-S
Storage Management Initiative Specification

4.6
SNIA
Storage Networking Industry Association

4.7
UFsT
User Friendly Selection Tag

5 Recipes

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

- smResetRSC
- smShowInstance
- smShowInstances
- smShowAssociationInstance
- smShowAssociationInstances
- smStartRSC
- smStopRSC

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

Table 1 – Local Recipes

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

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

5.1 lAddReferencedProperties

5.1.1 Description

Add the relevant associated properties to the instance of CIM_LogicalDevice.

5.1.2 Preconditions

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

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

5.1.3 Pseudo Code

```
sub lAddReferencedProperties($instance, #ReferencedPropertyNames[]){
   #propertylist[] = NULL;
//find the associated service, call the method as a query, insert the time as a
   referenced property
// Find the TimeService associated to the target system
#Error = smOpAssociators(
   $instance.getObjectPath(),
   “CIM_ServiceAffectsElement”,
   “CIM_TimeService”,
   NULL,
   NULL,
   $Services[]);
//if there is not a service associated the function is not supported.
if (0 == $Services[].length){
   //not supported, don't add property
      return;
}
$Service-> = $Services[0].getObjectPath();
 %InArguments[] = { newArgument(“GetRequest”, FALSE)
                    newArgument(“ManagedElement”, $instance.getOjectPath())}
%OutArguments[] = { newArgument(“TimeData”, #timedata)};
//invoke method
#returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
   “ManageTime”,
   %InArguments[],
   %OutArguments[]);

// process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if (#Error.code == 17)   {
      //not supported, so don't add property
      return;
   }
   else {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance(“CIM_Error”);
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
```

```
          $OperationError.ErrorType = 4;
          //Unknown
          $OperationError.PerceivedSeverity = 0;
          $OperationError.OwningEntity = DMTF:SMCLP;
          $OperationError.MessageID = 0x00000009;
          $OperationError.Message = “An internal software error has occurred.”;
          &smAddError($job, $OperationError);
          &smMakeCommandStatus($job);
          &smEnd;
       }
}//if CIM op failed
else if (0 == #returnStatus)  {
   //completed successfully
   $instance.CurrentTime = #timedata;
   #ReferencedPropertyNames = {“CurrentTime”};
}
else if (1 == #returnStatus)  {
   //not supported, so don't add
   return;
}
else {
   //generic failure
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Other
   $OperationError.ErrorType = 1;
   //Low
   $OperationError.PerceivedSeverity = 2;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000002;
   $OperationError.Message = “Failed. No further information is available.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
}
} //lAddReferencedProperties()
```

6 Mappings

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

6.1 CIM_ComputerSystem

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

Table 2 lists each SM CLP verb, the required level of support for the verb in conjunction with the target class, and, when appropriate, a cross-reference to the section detailing the mapping for the verb and 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`*****`>;`**

6.1.2.3.2 Pseudo Code

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

6.1.3 Set

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

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

6.1.3.1 General Usage of Set for a Single Property

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

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

6.1.3.1.1 Command Form

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

6.1.3.1.2 CIM Requirements

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

6.1.3.1.3 Behavior Requirements

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

6.1.3.2 General Usage of Set for Multiple Properties

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

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

6.1.3.2.1 Command Form

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

6.1.3.2.2 CIM Requirements

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

6.1.3.2.3 Behavior Requirements

```
$instance=<CIM_ComputerSystem single instance>
#propertyNames[] = {<propertyname>};
for #i < n
   {
      #propertyNames[#i] = <propertname#i>
      #propertyValues[#i] = <propertyvalue#i>
   }
&smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
&smEnd;
```

6.1.3.3 Set RequestedState to “Offline”

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

6.1.3.3.1 Command Form

```
set <CIM_ComputerSystem single instance> RequestedState=“Offline”
```

6.1.3.3.2 CIM Requirements

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

6.1.3.3.3 Behavior Requirements

6.1.3.3.3.1 Preconditions

`$instance` represents the targeted instance of CIM_ComputerSystem.

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

6.1.3.3.3.2 Pseudo Code

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

6.1.3.4 Set Service Processor Time

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

6.1.3.4.1 Command Form

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

6.1.3.4.2 CIM Requirements

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

6.1.3.4.3 Behavior Requirements

6.1.3.4.3.1 Preconditions

```
// #requestedtime contains the datetime which corresponds to the value of the
// currenttime or currenttime#time property value
// $instance=<CIM_ComputerSystem Single Instance>
// Find the TimeService associated to the target system
#Error = smOpAssociators(
   $instance.getObjectPath(),
   “CIM_ServiceAffectsElement”,
   “CIM_TimeService”,
   NULL,
   NULL,
   $Services[]);
//if there is not a service associated the function is not supported.
if (0 == $Services[].length){
       //unsupported
       $OperationError = smNewInstance(“CIM_Error”);
       //CIM_ERR_NOT_SUPPORTED
       $OperationError.CIMStatusCode = 7;
       //Other
       $OperationError.ErrorType = 1;
       //Low
       $OperationError.PerceivedSeverity = 2;
       $OperationError.OwningEntity = DMTF:SMCLP;
       $OperationError.MessageID = 0x00000001;
       $OperationError.Message = “Operation is not supported.”;
       &smAddError($job, $OperationError);
       &smMakeCommandStatus($job);
       &smEnd;
}
$Service-> = $Services[0].getObjectPath();
%InArguments[] = { newArgument(“GetRequest”, FALSE)
                   newArgument(“TimeData”, #time),
                   newArgument(“ManagedElement”, $instance.getOjectPath())}
%OutArguments[] = { };
//invoke method
#returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
   ”ManageTime”,
   %InArguments[],
   %OutArguments[]);
// process return code to CLP Command Status
if (0 != #Error.code)  {
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
      // if the method invocation contains an embedded error
      // use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
```

```
      &smEnd;
   }
   else if (#Error.code == 17)   {
   //trap for CIM_METHOD_NOT_FOUND
   //and make nice Unsupported msg.
      //unsupported
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
         $OperationError = smNewInstance(“CIM_Error”);
         //CIM_ERR_FAILED
         $OperationError.CIMStatusCode = 1;
         //Software Error
         $OperationError.ErrorType = 4;
         //Unknown
         $OperationError.PerceivedSeverity = 0;
         $OperationError.OwningEntity = DMTF:SMCLP;
         $OperationError.MessageID = 0x00000009;
         $OperationError.Message = “An internal software error has occurred.”;
         &smAddError($job, $OperationError);
         &smMakeCommandStatus($job);
         &smEnd;
   }
}//if CIM op failed
else if (0 == #returnStatus)  {
   //completed successfully
   %InArguments[] = { newArgument(“GetRequest”, FALSE)
                      newArgument(“ManagedElement”, $instance.getOjectPath())}
   %OutArguments[] = { newArgument(“TimeData”, #timedata)};
//invoke method
   #returnStatus = smOpInvokeMethod ($Service.GetObjectPath(),
      ”ManageTime”,
      %InArguments[],
      %OutArguments[]);
   if (0 != #Error.code)  {
```

```
      //method invocation failed
      if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )    {
         // if the method invocation contains an embedded error
         // use it for the Error for the overall job
         &smAddError($job, #Error.$error[0]);
         &smMakeCommandStatus($job);
         &smEnd;
      }
   }
   else {
      //generic failure
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000002;
      $OperationError.Message = “Failed. No further information is available.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
   }
   &smEnd;
}
else if (1 == #returnStatus)  {
      //unsupported
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_NOT_SUPPORTED
      $OperationError.CIMStatusCode = 7;
      //Other
      $OperationError.ErrorType = 1;
      //Low
      $OperationError.PerceivedSeverity = 2;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
}
else {
   //generic failure
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Other
   $OperationError.ErrorType = 1;
```

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

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[] );
```

```
if (0 != #Error.code)
   {
   &smProcessOpError (#Error);
   //includes &smEnd;
   }

// Step 2 - add their referenced properties
for $instance in $instances[] {
   &lAddReferencedProperties (
   $instance,
   #referencedPropertyNames[] );
   }

//step 3 – display them
&smShowInstancesWithReferencedProperties (
   $instances[],
   #propertyList[],
   #referencedPropertyNames[] );

&smEnd;
```

6.1.4.2 Show Command Form for a Single Instance Target

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

6.1.4.2.1 Command Form

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

6.1.4.2.2 CIM Requirements

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

6.1.4.2.3 Behavior Requirements

6.1.4.2.3.1 Preconditions

`$instance` represents the targeted instance of CIM_ComputerSystem.

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

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

6.1.4.2.3.2 Pseudo Code

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

6.1.5 Start

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

6.1.5.1 Command Form

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

6.1.5.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
uint32 CIM_ComputerSystem.RequestStateChange (
   [IN] uint16 RequestedState,
   [OUT] REF CIM_ConcreteJob Job,
   [IN] datetime TimeoutPeriod );
```

6.1.5.3 Behavior Requirements

6.1.5.3.1 Preconditions

`$instance` represents the targeted instance of CIM_ComputerSystem.

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

6.1.5.3.1.1 Pseudo Code

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

6.1.6 Stop

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

6.1.6.1 Command Form

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

6.1.6.2 CIM Requirements

```
uint16 EnabledState;
uint16 RequestedState;
   uint32 CIM_ComputerSystem.RequestStateChange (
   [IN] uint16 RequestedState,
   [OUT] REF CIM_ConcreteJob Job,
   [IN] datetime TimeoutPeriod );
```

6.1.6.3 Behavior Requirements

6.1.6.3.1.1 Preconditions

`$instance` represents the targeted instance of CIM_ComputerSystem.

```
$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>
```

6.2.2.1.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.2.2.1.3 Behavior Requirements

6.2.2.1.3.1 Preconditions

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

6.2.2.1.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getObjectPath() );
&smEnd;
```

6.2.2.2 Show Command Form for a Single Instance – CIM_ComputerSystem Reference

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

6.2.2.2.1 Command Form

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

6.2.2.2.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.2.2.2.3 Behavior Requirements

6.2.2.2.3.1 Preconditions

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

6.2.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_ElementCapabilities”, $instance.getObjectPath() );
&smEnd;
```

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

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

6.2.2.3.1 Command Form

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

6.2.2.3.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.2.2.3.3 Behavior Requirements

6.2.2.3.3.1 Preconditions

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

6.2.2.3.3.2 Pseudo Code

```
&smShowAssociationInstance ( “CIM_ElementCapabilities”, $instanceA.getObjectPath(),
   $instanceB.getObjectPath() );
&smEnd;
```

6.3 CIM_EnabledLogicalElementCapabilities

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

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

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	

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

6.3.1 Ordering of Results

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

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

6.3.2 Show

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

6.3.2.1 Show Command Form for Multiple Instances Target

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

6.3.2.1.1 Command Form

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

6.3.2.1.2 CIM Requirements

See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.3.2.1.3 Behavior Requirements

6.3.2.1.3.1 Preconditions

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

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

6.3.2.1.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowInstances ( “CIM_EnabledLogicalElementCapabilities”, “CIM_MemberOfCollection”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.3.2.2 Show Command Form for a Single Instance Target

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

6.3.2.2.1 Command Form

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

6.3.2.2.2 CIM Requirements

See CIM_EnabledLogicalElementCapabilities in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.3.2.2.3 Behavior Requirements

6.3.2.2.3.1 Preconditions

`$instance` represents the targeted instance of CIM_EnabledLogicalElementCapabilities.

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

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

6.3.2.2.3.2 Pseudo Code

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

6.4 CIM_HostedService

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

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

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	

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

6.4.1 Ordering of Results

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

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

6.4.2 Show

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

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

6.4.2.1 Show Multiple Instances

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

6.4.2.1.1 Command Form

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

6.4.2.1.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 Preconditions

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

6.4.2.1.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_HostedService”, $instance.getObjectPath() );
&smEnd;
```

6.4.2.2 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_HostedService where the reference specified is to an instance of CIM_TimeService. An instance of CIM_TimeService is referenced by exactly one instance of CIM_HostedService. Therefore, a single instance will be returned.

6.4.2.2.1 Command Form

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

6.4.2.2.2 CIM Requirements

See CIM_HostedService in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

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

6.4.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_HostedService”, $instance.getObjectPath() );
&smEnd;
```

6.4.2.3 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_HostedService where both references are specified and therefore the desired instance is unambiguously identified.

6.4.2.3.1 Command Form

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

6.4.2.3.2 CIM Requirements

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

6.4.2.3.3 Behavior Requirements

6.4.2.3.3.1 Preconditions

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

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

6.4.2.3.3.2 Pseudo Code

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

6.5 CIM_IsSpare

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

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

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	

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

6.5.1 Ordering of Results

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

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

6.5.2 Show

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

6.5.2.1 Show Command Form for Multiple Instances Target – CIM_RedundancySet Reference

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

6.5.2.1.1 Command Form

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

6.5.2.1.2 CIM Requirements

See CIM_IsSpare in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.5.2.1.3 Behavior Requirements

6.5.2.1.3.1 Preconditions

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

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

6.5.2.1.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_IsSpare”, $instance.getObjectPath(),
   #propertylist[] );
&smEnd;
```

6.5.2.2 Show Command Form for a Single Instance – CIM_ComputerSystem Reference

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

6.5.2.2.1 Command Form

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

6.5.2.2.2 CIM Requirements

See CIM_IsSpare in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.5.2.2.3 Behavior Requirements

6.5.2.2.3.1 Preconditions

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

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

6.5.2.2.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstances ( “CIM_IsSpare”, $instance.getObjectPath(),
   #propertylist[]);
&smEnd;
```

6.5.2.3 Show Command Form for a Single Instance Target – 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_IsSpare where both references are specified and therefore the desired instance is unambiguously identified.

6.5.2.3.1 Command Form

`show <CIM_IsSpare `*`single instance`*`>`

6.5.2.3.2 CIM Requirements

See CIM_IsSpare in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.5.2.3.3 Behavior Requirements

6.5.2.3.3.1 Preconditions

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

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

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

6.5.2.3.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowAssociationInstance ( “CIM_IsSpare”, $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.6 CIM_MemberOfCollection

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

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

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	

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

6.6.1 Ordering of Results

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

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

6.6.2 Show

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

6.6.2.1 Show Command Form for Multiple Instances Target – CIM_RedundancySet Reference

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

6.6.2.1.1 Command Form

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

6.6.2.1.2 CIM Requirements

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

6.6.2.1.3 Behavior Requirements

6.6.2.1.3.1 Preconditions

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

6.6.2.1.3.2 Pseudo Code

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

6.6.2.2 Show Command Form for a Single Instance – CIM_ComputerSystem Reference

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

6.6.2.2.1 Command Form

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

6.6.2.2.2 CIM Requirements

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

6.6.2.2.3 Behavior Requirements

6.6.2.2.3.1 Preconditions

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

6.6.2.2.3.2 Pseudo Code

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

6.6.2.3 Show Command Form for a Single Instance Target – 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_MemberOfCollection where both references are specified and therefore the desired instance is unambiguously identified.

6.6.2.3.1 Command Form

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

6.6.2.3.2 CIM Requirements

See CIM_MemberOfCollection in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.6.2.3.3 Behavior Requirements

6.6.2.3.3.1 Preconditions

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

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

6.6.2.3.3.2 Pseudo Code

```
&smShowAssociationInstance ( “CIM_MemberOfCollection”, $instanceA.getObjectPath(),
   $instanceB.getObjectPath() );
&smEnd;
```

6.7 CIM_OwningCollectionElement

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

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

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	

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

6.7.2.2.3 Behavior Requirements

6.7.2.2.3.1 Preconditions

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

6.7.2.2.3.2 Pseudo Code

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

6.7.2.3 Show Command Form for a Single Instance Target – 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_OwningCollectionElement where both references are specified and therefore the desired instance is unambiguously identified.

6.7.2.3.1 Command Form

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

6.7.2.3.2 CIM Requirements

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

6.7.2.3.3 Behavior Requirements

6.7.2.3.3.1 Preconditions

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

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

6.7.2.3.3.2 Pseudo Code

```
&smShowAssociationInstance ( “CIM_OwningCollectionElement”,
   $instanceA.getObjectPath(), $instanceB.getObjectPath() );
&smEnd;
```

6.8 CIM_RedundancySet

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

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

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

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

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

1097 ### 6.8.1 Ordering of Results

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

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

1101 ### 6.8.2 Set

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

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

1105 #### 6.8.2.1 Set Command for Failovers

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

1108 ##### 6.8.2.1.1 Command Form

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

1111 ##### 6.8.2.1.2 CIM Requirements

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

1115 ##### 6.8.2.1.3 Behavior Requirements

1116 ###### 6.8.2.1.3.1 Preconditions

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

6.8.2.1.3.2 Pseudo Code

```
%InArguments[] = { newArgument ( “FailoverFrom”, $FailoverFrom.getObjectPath() ),
       newArgument ( “FailoverTo”, $FailoverTo.getObjectPath () };
%OutArguments[] = {};
#Error = InvokeMethod ($target->,
                       “Failover”,
                       %InArguments[],
                       %OutArguments[],
                       #returnStatus);
if (0 != #Error.code)
{
   //method invocation failed
   if ( (NULL != #Error.$error) && (NULL != #Error.$error[0]) )
   {
      //if the method invocation contains an embedded error
      //use it for the Error for the overall job
      &smAddError($job, #Error.$error[0]);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else if ( 17 == #Error.code ) {
      //17 – CIM_ERR_METHOD_NOT_FOUND
      // The specified extrinsic method does not exist.
      $OperationError = smNewInstance(“CIM_Error”);
      // CIM_ERR_METHOD_NOT_FOUND
      $OperationError.CIMStatusCode = 17;
      //Software Error
      $OperationError.ErrorType = 10;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
      $OperationError.MessageID = 0x00000001;
      $OperationError.Message = “Operation is not supported.”
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
   else
   {
      //operation failed, but no detailed error instance, need to make one up
      //make an Error instance and associate with job for Operation
      $OperationError = smNewInstance(“CIM_Error”);
      //CIM_ERR_FAILED
      $OperationError.CIMStatusCode = 1;
      //Software Error
      $OperationError.ErrorType = 4;
      //Unknown
      $OperationError.PerceivedSeverity = 0;
      $OperationError.OwningEntity = DMTF:SMCLP;
```

```
      $OperationError.MessageID = 0x00000009;
      $OperationError.Message = “An internal software error has occurred.”;
      &smAddError($job, $OperationError);
      &smMakeCommandStatus($job);
      &smEnd;
   }
}//if CIM op failed
else if (0 == #returnStatus)  {
   //completed successfully
   &smCommandCompleted($job);
   &smEnd;
}
else if (1 == #returnStatus)  {
   //unsupported
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_NOT_SUPPORTED
   $OperationError.CIMStatusCode = 7;
   //Other
   $OperationError.ErrorType = 1;
   //Low
   $OperationError.PerceivedSeverity = 2;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000001;
   $OperationError.Message = “Operation is not supported.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
   &smEnd;
}
else if (2 == #returnStatus)  {
   //generic failure
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Other
   $OperationError.ErrorType = 1;
   //Low
   $OperationError.PerceivedSeverity = 2;
   $OperationError.OwningEntity = DMTF:SMCLP;
   $OperationError.MessageID = 0x00000002;
   $OperationError.Message = “Failed. No further information is available.”;
   &smAddError($job, $OperationError);
   &smMakeCommandStatus($job);
}
else   {
   //unspecified return code, generic failure
   $OperationError = smNewInstance(“CIM_Error”);
   //CIM_ERR_FAILED
   $OperationError.CIMStatusCode = 1;
   //Other
```

```
    $OperationError.ErrorType = 1;
    //Low
    $OperationError.PerceivedSeverity = 2;
    $OperationError.OwningEntity = DMTF:SMCLP;
    $OperationError.MessageID = 0x00000002;
    $OperationError.Message = "Failed. No further information is available.";
    &smAddError($job, $OperationError);
    &smMakeCommandStatus($job);
    &smEnd;
}
```

6.8.2.2 General Usage of Set for a Single Property

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

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

6.8.2.2.1 Command Form

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

6.8.2.2.2 CIM Requirements

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

6.8.2.2.3 Behavior Requirements

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

6.8.2.3 General Usage of Set for Multiple Properties

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

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

6.8.2.3.1 Command Form

`set <CIM_RedundancySet `*`single instance`*`> <propertyname1>=<propertyvalue1>`
`    <propertyname`*`n`*`>=<propertyvalue`*`n`*`>`

6.8.2.3.2 CIM Requirements

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

6.8.2.3.3 Behavior Requirements

```
$instance=<CIM_RedundancySet single instance>
#propertyNames[] = {<propertyname>};
for #i < n
   {
   #propertyNames[#i] = <propertname#i>
   #propertyValues[#i] = <propertyvalue#i>
   }
&smSetInstance ( $instance, #propertyNames[], #propertyValues[] );
&smEnd;
```

6.8.3 Show

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

6.8.3.1 Show Command Form for Multiple Instances Target

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

6.8.3.1.1 Command Form

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

6.8.3.1.2 CIM Requirements

See CIM_RedundancySet in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.8.3.1.3 Behavior Requirements

6.8.3.1.3.1 Preconditions

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

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

6.8.3.1.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all )
   {
   #propertylist[] = <array of mandatory non-key property names (see CIM
      Requirements)>;
   }
&smShowInstances ( “CIM_RedundancySet”, “CIM_OwningCollectionElement”,
   $containerInstance.getObjectPath(), #propertylist[] );
&smEnd;
```

6.8.3.2 Show Command Form for a Single Instance Target

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

6.8.3.2.1 Command Form

```
show <CIM_RedundancySet single instance>
```

6.8.3.2.2 CIM Requirements

See CIM_RedundancySet in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.8.3.2.3 Behavior Requirements

6.8.3.2.3.1 Preconditions

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

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

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

6.8.3.2.3.2 Pseudo Code

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

6.9 CIM_ServiceAffectsElement

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

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

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	

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

6.9.2.3.3.2 Pseudo Code

```
#propertylist[] = NULL;
if ( false == #all )
   {
      #propertylist[] = {//all mandatory non-key properties};
   }
&smShowAssociationInstance ( “CIM_ServiceAffectsElement”, $instanceA.getObjectPath(),
   $instanceB.getObjectPath(), #propertylist[] );
&smEnd;
```

6.10 CIM_TimeService

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

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

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	

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

6.10.1 Ordering of Results

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

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

6.10.2 Show

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

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

6.10.2.1 Show a Single Instance

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

6.10.2.1.1 Command Form

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

6.10.2.1.2 CIM Requirements

See CIM_TimeService in the “CIM Elements” section of the *Service Processor Profile* for the list of mandatory properties.

6.10.2.1.3 Behavior Requirements

6.10.2.1.3.1 Preconditions

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

6.10.2.1.3.2 Pseudo Code

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

ANNEX A
(informative)

Change Log

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