Document Number: DSP0814

Date: 2009-07-14

Version: 1.0.0

Fan Profile SM CLP Command Mapping Specification

Document Type: Specification

Document Status: DMTF Standard

Document Language: E

CONTENTS

Tables

66

Foreword

The *Fan Profile SM CLP Command Mapping Specification* (DSP0814) was prepared by the Server Management Working Group.

Conventions

The pseudo-code conventions utilized in this document are the Recipe Conventions as defined in the 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:

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

Introduction

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

The target audience for this specification is implementers of the SM CLP support for the *Fan Profile*.

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

DMTF DSP1013, *Fan Profile 1.0*,
http://www.dmtf.org/standards/published_documents/DSP1013_1.0.pdf

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

2.2 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
IETF
Internet Engineering Task Force

4.5
SM
Server Management

4.6
SMI-S
Storage Management Initiative Specification

4.7
SNIA
Storage Networking Industry Association

4.8
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 the *SM CLP-to-CIM Common Mapping Specification 1.0* (DSP0216).

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

6 Mappings

The following sections detail the mapping of CLP verbs to CIM Operations for each CIM class defined in *Fan 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_AssociatedCooling

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

Table 1 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 1 is for informational purposes only; in case of a conflict between Table 1 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 1.

197 **Table 1 – Command Verb Requirements for CIM_AssociatedCooling**

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 6.8.2.
Start	Not supported	
Stop	Not supported	

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

200 ### 6.1.1 Ordering of Results

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

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

204 ### 6.1.2 Show

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

208 #### 6.1.2.1 Show Command Form for Multiple Instances Target – CIM_ManagedSystemElement
209 Reference

210 This command form is used to show many instances of CIM_AssociatedCooling. This command form
211 corresponds to a `show` command issued against the instance of CIM_AssociatedCooling where only one
212 reference is specified and the reference is to the instance of CIM_ManagedSystemElement.

213 ##### 6.1.2.1.1 Command Form

214 **`show <CIM_AssociatedCooling` *`multiple instances`*`>`**

215 ##### 6.1.2.1.2 CIM Requirements

216 See CIM_AssociatedCooling in the “CIM Elements” section of the *Fan Profile* for the list of mandatory
217 properties.

218 ##### 6.1.2.1.3 Behavior Requirements

219 ###### 6.1.2.1.3.1 Preconditions

220 `$instance` represents the instance of a CIM_ManagedSystemElement, which is referenced by
221 CIM_AssociatedCooling.

6.1.2.1.3.2 Pseudo Code

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

6.1.2.2 Show Command Form for Multiple Instance Target – CIM_Fan Reference

This command form is used to show multiple instances of CIM_AssociatedCooling. This command form corresponds to a `show` command issued against multiple instances of CIM_AssociatedCooling, where only one reference is specified and the reference is to the instance of CIM_Fan.

6.1.2.2.1 Command Form

```
show <CIM_AssociatedCooling multiple instances>
```

6.1.2.2.2 CIM Requirements

See CIM_AssociatedCooling in the “CIM Elements” section of the *Fan Profile* for the list of mandatory properties.

6.1.2.2.3 Behavior Requirements

6.1.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_Fan which is referenced by CIM_AssociatedCooling.

6.1.2.2.3.2 Pseudo Code

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

6.1.2.3 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_AssociatedCooling where both references are specified and therefore the desired instance is unambiguously identified.

6.1.2.3.1 Command Form

```
show <CIM_AssociatedCooling single instance>
```

6.1.2.3.2 CIM Requirements

See CIM_AssociatedCooling in the “CIM Elements” section of the *Fan Profile* for the list of mandatory properties.

6.1.2.3.3 Behavior Requirements

6.1.2.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_Fan through CIM_AssociatedCooling association.

`$instanceB` represents the instance of CIM_ManagedSystemElement which is referenced by CIM_AssociatedCooling.

6.1.2.3.3.2 Pseudo Code

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

6.2 CIM_ElementCapabilities

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

Table 2 – 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	

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

6.2.1 Ordering of Results

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

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

6.2.2 Show

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

6.2.2.1 Show Command Form for Multiple Instances Target – CIM_EnabledLogicalElementCapabilities Reference

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

6.2.2.1.1 Command Form

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

6.2.2.1.2 CIM Requirements

See CIM_ElementCapabilities in the “CIM Elements” section of the *Fan 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_Fan 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_Fan.

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 *Fan 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_Fan 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 a `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 *Fan 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_Fan through 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 3 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 3 is for informational purposes only; in case of a conflict between Table 3 and requirements detailed in the following sections, the text detailed in the following sections supersedes the information in Table 3.

Table 3 – 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 *Fan 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 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 *Fan 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_IsSpare

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_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 6.4.2.
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`.

6.4.1 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.

6.4.2 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.

6.4.2.1 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.

6.4.2.1.1 Command Form

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

6.4.2.1.2 CIM Requirements

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

6.4.2.1.3 Behavior Requirements

6.4.2.1.3.1 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.

6.4.2.1.3.2 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;
```

6.4.2.2 Show Command Form for a Single Instance – CIM_Fan 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_Fan.

6.4.2.2.1 Command Form

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

6.4.2.2.2 CIM Requirements

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

6.4.2.2.3 Behavior Requirements

6.4.2.2.3.1 Preconditions

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

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

6.4.2.2.3.2 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;
```

6.4.2.3 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.

6.4.2.3.1 Command Form

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

6.4.2.3.2 CIM Requirements

See CIM_IsSpare in the "CIM Elements" section of the *Fan Profile* for the list of mandatory properties.

6.4.2.3.3 Behavior Requirements

6.4.2.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_Fan through 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.

6.4.2.3.3.2 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;
```

6.5 CIM_MemberOfCollection

The `cd`, `exit`, `help`, and `version` 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 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_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 6.5.2.
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_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.

6.5.2 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.

6.5.2.1 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.

6.5.2.1.1 Command Form

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

6.5.2.1.2 CIM Requirements

See CIM_MemberOfCollection in the “CIM Elements” section of the *Fan 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_MemberOfCollection.

6.5.2.1.3.2 Pseudo Code

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

6.5.2.2 Show Command Form for a Single Instance – CIM_Fan 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_Fan.

6.5.2.2.1 Command Form

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

6.5.2.2.2 CIM Requirements

See CIM_MemberOfCollection in the “CIM Elements” section of the *Fan 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_Fan which is referenced by CIM_MemberOfCollection.

6.5.2.2.3.2 Pseudo Code

```
&smShowAssociationInstances ( “CIM_MemberOfCollection”, $instance.getObjectPath() );
&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_MemberOfCollection where both references are specified and therefore the desired instance is unambiguously identified.

6.5.2.3.1 Command Form

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

6.5.2.3.2 CIM Requirements

See CIM_MemberOfCollection in the “CIM Elements” section of the *Fan 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_Fan through CIM_MemberOfCollection association.

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

6.5.2.3.3.2 Pseudo Code

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

6.6 CIM_Fan

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_Fan

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	May	See 6.6.2.
Set	May	See 6.6.3.
Show	Shall	See 6.6.4.
Start	May	See 6.6.5.
Stop	May	See 6.6.6.

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

6.6.1 Ordering of Results

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

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

6.6.2 Reset

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

6.6.2.1 Command Form

```
reset <CIM_Fan single instance>
```

6.6.2.2 CIM Requirements

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

6.6.2.3 Behavior Requirements

6.6.2.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Fan.

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

6.6.2.3.1.1 Pseudo Code

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

6.6.3 Set

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

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

6.6.3.1 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 *Fan Profile*.

6.6.3.1.1 Command Form

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

6.6.3.1.2 CIM Requirements

See CIM_Fan in the “CIM Elements” section of the *Fan Profile* for the list of mandatory properties.

6.6.3.1.3 Behavior Requirements

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

6.6.3.2 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 *Fan Profile*.

6.6.3.2.1 Command Form

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

6.6.3.2.2 CIM Requirements

See CIM_Fan in the “CIM Elements” section of the *Fan Profile* for the list of mandatory properties.

6.6.3.2.3 Behavior Requirements

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

6.6.3.3 Setting the DesiredSpeed Property

This section describes how to set the speed of a fan.

6.6.3.3.1 Command Form

```
set <CIM_Fan single instance> desiredspeed=<requested value>
```

6.6.3.3.2 CIM Requirements

```
uint64 DesiredSpeed;
uint32 CIM_Fan.SetSpeed (
   [IN] uint64 DesiredSpeed );
```

6.6.3.3.3 Behavior Requirements

6.6.3.3.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Fan.

```
$instance=<CIM_Fan single instance>;
#desiredSpeed=<requested value>
```

6.6.3.3.3.2 Pseudo Code

```
%InArguments[] = {newArgument(“DesiredSpeed”, #desiredSpeed) };
                        %OutArguments[] = {};
#Error = InvokeMethod ($instance.getObjectPath(),
                        “SetSpeed”,
                        %InArguments[],
                        %OutArguments[],
                        #returnStatus);
if (0 != #Error.code)  {
//method invocation failed
   if ( (null != #Error.$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   {
       // 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;
}
&smDisplayInstance ( $instance );
&smEnd;
```

6.6.4 Show

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

6.6.4.1 Show Command Form for Multiple Instances Target

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

6.6.4.1.1 Command Form

```
show <CIM_Fan multiple instances>
```

6.6.4.1.2 CIM Requirements

See CIM_Fan in the "CIM Elements" section of the *Fan Profile* for the list of mandatory properties.

6.6.4.1.3 Behavior Requirements

6.6.4.1.3.1 Preconditions

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

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

6.6.4.1.3.2 Pseudo Code

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

6.6.4.2 Show Command Form for a Single Instance Target

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

6.6.4.2.1 Command Form

```
show <CIM_Fan single instance>
```

6.6.4.2.2 CIM Requirements

See CIM_Fan in the “CIM Elements” section of the *Fan Profile* for the list of mandatory properties.

6.6.4.2.3 Behavior Requirements

6.6.4.2.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Fan.

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

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

6.6.4.2.3.2 Pseudo Code

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

6.6.5 Start

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

6.6.5.1.1 Command Form

```
start <CIM_Fan single instance>
```

6.6.5.1.2 CIM Requirements

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

6.6.5.1.3 Behavior Requirements

6.6.5.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Fan.

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

6.6.5.1.3.2 Pseudo Code

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

6.6.6 Stop

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

6.6.6.1.1 Command Form

```
stop <CIM_Fan single instance>
```

6.6.6.1.2 CIM Requirements

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

6.6.6.1.3 Behavior Requirements

6.6.6.1.3.1 Preconditions

`$instance` represents the targeted instance of CIM_Fan.

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

6.6.6.1.3.2 Pseudo Code

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

6.7 CIM_RedundancySet

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_RedundancySet

Command Verb	Requirement	Comments
Create	Not supported	
Delete	Not supported	
Dump	Not supported	
Load	Not supported	
Reset	Not supported	
Set	May	See 6.7.2.
Show	Shall	See 6.7.3.
Start	Not supported	
Stop	Not supported	

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

6.7.1 Ordering of Results

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

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

6.7.2 Set

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

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

6.7.2.1 Set Command for Failovers

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

6.7.2.1.1 Command Form

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

6.7.2.1.2 CIM Requirements

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

6.7.2.1.3 Behavior Requirements

6.7.2.1.3.1 Preconditions

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

6.7.2.1.3.2 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;
}
```

6.7.2.2 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 *Fan Profile*.

6.7.2.2.1 Command Form

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

6.7.2.2.2 CIM Requirements

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

6.7.2.2.3 Behavior Requirements

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

6.7.2.3 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 *Fan Profile*.

6.7.2.3.1 Command Form

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

6.7.2.3.2 CIM Requirements

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

6.7.2.3.3 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;
```

6.7.3 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.

6.7.3.1 Show Command Form for Multiple Instances Target

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

6.7.3.1.1 Command Form

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

6.7.3.1.2 CIM Requirements

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

6.7.3.1.3 Behavior Requirements

6.7.3.1.3.1 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.

6.7.3.1.3.2 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;
```

6.7.3.2 Show Command Form for a Single Instance Target

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

6.7.3.2.1 Command Form

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

6.7.3.2.2 CIM Requirements

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

6.7.3.2.3 Behavior Requirements

6.7.3.2.3.1 Preconditions

`$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.

6.7.3.2.3.2 Pseudo Code

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

6.8 CIM_SystemDevice

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_SystemDevice

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 6.8.2.
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`.

6.8.1 Ordering of Results

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

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

6.8.2 Show

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

6.8.2.1 Show Command Form for Multiple Instances Target – CIM_ComputerSystem Reference

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

6.8.2.1.1 Command Form

```
show <CIM_SystemDevice multiple instances>
```

6.8.2.1.2 CIM Requirements

See CIM_SystemDevice in the "CIM Elements" section of the *Fan Profile* for the list of mandatory properties.

6.8.2.1.3 Behavior Requirements

6.8.2.1.3.1 Preconditions

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

6.8.2.1.3.2 Pseudo Code

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

6.8.2.2 Show Command Form for a Single Instance Target – CIM_Fan Reference

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

6.8.2.2.1 Command Form

```
show <CIM_SystemDevice single instance>
```

6.8.2.2.2 CIM Requirements

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

6.8.2.2.3 Behavior Requirements

6.8.2.2.3.1 Preconditions

`$instance` represents the instance of CIM_Fan which is referenced by the CIM_SystemDevice association.

6.8.2.2.3.2 Pseudo Code

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

6.8.2.3 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_SystemDevice where both references are specified and therefore the desired instance is unambiguously identified.

6.8.2.3.1 Command Form

```
show <CIM_SystemDevice single instance>
```

6.8.2.3.2 CIM Requirements

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

6.8.2.3.3 Behavior Requirements

6.8.2.3.3.1 Preconditions

`$instanceA` represents the referenced instance of CIM_Fan through CIM_SystemDevice association.

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

6.8.2.3.3.2 Pseudo Code

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

6.9 CIM_OwningCollectionElement

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.

Table 9 – 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 6.8.2.
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`.

6.9.1 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.

6.9.2 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.

6.9.2.1 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.

6.9.2.1.1 Command Form

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

6.9.2.1.2 CIM Requirements

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

6.9.2.1.3 Behavior Requirements

6.9.2.1.3.1 Preconditions

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

6.9.2.1.3.2 Pseudo Code

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

6.9.2.2 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.

6.9.2.2.1 Command Form

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

6.9.2.2.2 CIM Requirements

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

6.9.2.2.3 Behavior Requirements

6.9.2.2.3.1 Preconditions

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

6.9.2.2.3.2 Pseudo Code

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

6.9.2.3 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.

6.9.2.3.1 Command Form

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

6.9.2.3.2 CIM Requirements

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

6.9.2.3.3 Behavior Requirements

6.9.2.3.3.1 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.

6.9.2.3.3.2 Pseudo Code

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

ANNEX A
(informative)

Change Log

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